IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF DELAWARE

IPA TECHNOLOGIES, INC.,	)	**PUBLIC VERSION**
Plaintiff,	)	The Honorable Richard G. Andrews
v.	)	
AMAZON.COM, INC., and AMAZON DIGITAL SERVICES, LLC,	)	Civil Action 1:16-CV-01266-RGA
Defendants.	)	**JURY TRIAL DEMANDED**

DECLARATION OF VIGEN SALMASTLIAN IN SUPPORT OF ANSWERING BRIEF IN OPPOSITION TO IPA TECHNOLOGIES, INC.'S MOTION FOR SUMMARY JUDGMENT OF NO INVALIDITY BASED ON THE OAA SYSTEM

(VOLUME II OF II)

Dated: July 30, 2021

Of Counsel:

J. David Hadden
Saina S. Shamilov
Ravi Ranganath
Vigen Salmastlian
FENWICK & WEST LLP
801 California Street Mountain View, CA 94041 (650) 988-8500

Todd R. Gregorian
Sapna S. Mehta
Eric B. Young
Min Wu
FENWICK & WEST LLP
555 California Street, 12th Floor
San Francisco, CA 94104 (415) 875-2300

ASHBY & GEDDES

Steven J. Balick (#2114)
Andrew C. Mayo (#5207)
500 Delaware Avenue, 8th Floor
P.O. Box 1150
Wilmington, DE 19899
(302) 654-1888
sbalick@ashbygeddes.com
amayo@ashbygeddes.com

Attorneys for Defendants Amazon.com, Inc. and Amazon Digital Services, LLC

I, Vigen Salmastlian, declare as follows:

1. I am a licensed attorney admitted in the State of California and admitted *pro hac vice* to this Court. I am an associate with the law firm of Fenwick & West LLP, counsel of record for defendants Amazon.com, Inc. and Amazon Digital Services LLC (collectively, "Amazon"). I have personal knowledge of the facts in this declaration and can competently testify to those facts.

2. Attached hereto as **Exhibit 1** is a true and correct copy of U.S. Patent No. 6,851,115.

3. Attached hereto as **Exhibit 2** is a true and correct copy of U.S. Patent No. 7,096,560.

4. Attached hereto as **Exhibit 3** is a true and correct copy of the Errata to Opening Expert Report of Katia P. Sycara, Ph.D. dated March 12, 2021.

5. Attached hereto as **Exhibit 4** is a true and correct copy of the declaration of David L. Martin filed in this case July 6, 2021.

6. Attached hereto as **Exhibit 5** is a true and correct copy of the declaration of Raymond C. Perrault filed in this case on July 6, 2021.

7. Attached hereto as **Exhibit 6** is a true and correct copy of the deposition transcript of Adam Cheyer in this case dated November 9, 2020.

8. Attached hereto as **Exhibit 7** is a true and correct copy of the deposition transcript of David Martin in this case dated November 11, 2020.

9. Attached hereto as **Exhibit 8** is a true and correct copy of the deposition transcript of Douglas Moran in this case dated November 16, 2020.

10. Attached hereto as **Exhibit 9** is a true and correct copy of the deposition transcript of Luc Julia in this case dated December 1, 2020.

11. Attached hereto as **Exhibits 10** is a true and correct copy of the declaration of Luc Julia produced in this case on December 3, 2020.

12. Attached hereto as **Exhibits 11** is a true and correct copy of the document subpoena served on SRI International in this case on July 18, 2019.

13. Attached hereto as **Exhibit 12** is a true and correct copy of the curriculum vitae of Douglas Moran, which is Exhibit 1 from the deposition of Douglas Moran in this case.

14. Attached hereto as **Exhibit 13** is a true and correct copy of the declaration of Douglas B. Moran from IPR2019-00728. *Google LLC v. IPA Techs. Inc.*, IPR2019-00728, Paper 1, Ex. 1007 (PTAB Feb. 26, 2019).

15. Attached hereto as **Exhibit 14** is a true and correct copy of the file [redacted]

16. Attached hereto as **Exhibit 15** is a true and correct copy of a May 16, 1997 internet archive capture of the webpage http://www.ai.sri.com/~oaa/distrib.html, which is Exhibit 26 from the deposition of Adam Cheyer in this case.

17. Attached hereto as **Exhibit 16** is a true and correct copy of the "OAA-USERS Mailing List Archive by date" from the webpage http://www.ai.sri.com/~oaa/distribution/MailArchive/oaa-users/date.html, which is Exhibit 31 from the deposition of Adam Cheyer in this case.

18. Attached hereto as **Exhibit 17** is a true and correct copy of Exhibit 6 to the declaration of Raymond C. Perrault filed July 6, 2021.

19. Attached hereto as **Exhibit 18** is a true and correct copy of the Reply Expert Report of Katia P. Sycara, Ph.D. dated May 7, 2021.

20. Attached hereto as **Exhibit 19** is a true and correct copy of the webpage http://www.adam.cheyer.com/demo-office.html, which is Exhibit 9 from the deposition of Adam Cheyer in this case.

21. Attached hereto as **Exhibit 20** is a true and correct copy of a video produced as DEFS_IPA_PA00021314 titled "Open Agent Architecture Automated Office Demonstration."

22. Attached hereto as **Exhibit 21** is a true and correct copy of [redacted]

23. Attached hereto as **Exhibit 22** is a true and correct copy of [redacted]

24. Attached hereto as **Exhibit 23** is a true and correct copy of [redacted]

25. Attached hereto as **Exhibit 24** is a true and correct copy of a publication produced as DEFS_IPA_PA00034264-279, titled "MVIEWS: Multimodal Tools for the Video Analyst," which is Exhibit 4 from the deposition of Adam Cheyer.

26. Attached hereto as **Exhibit 25** is a true and correct copy of a presentation produced as DEFS_IPA_PA00021059-1136, titled "PAAM '98 Tutorial Building and Using Practical Agent Applications."

27. Attached hereto as **Exhibit 26** is a true and correct copy of the deposition transcript of Adam Cheyer from IPR2019-00728. *Google LLC v. IPA Technologies Inc.*, IPR2019-00728, Paper 2037 (PTAB December 17, 2019).

28. Attached hereto as **Exhibit 27** is a true and correct copy of the deposition transcript of David Martin from IPR2019-00728. *Google LLC v. IPA Technologies Inc.*, IPR2019-00728, Paper 2038 (PTAB December 18, 2019).

29. Attached hereto as **Exhibit 28** is a true and correct copy of an excerpt of the certified prosecution history of U.S. Patent No. 6,851,115.

30. [redacted]

31. [redacted]

32. Reproduced below is a true and correct excerpt of the [redacted]

33. Reproduced below are true and correct [redacted]

I declare under penalty of perjury under the laws of the United States of America that the foregoing is true and correct. Executed in Saratoga, California on July 30, 2021.

/s/ Vigen Salmastlian
Vigen Salmastlian

Exhibit 11

Fenwick
FENWICK & WEST LLP
SILICON VALLEY CENTER 801 CALIFORNIA STREET MOUNTAIN VIEW, CA 94041
TEL 650.988.8500 FAX 650.938.5200 WWW.FENWICK.COM

July 18, 2019

VIGEN SALMASTLIAN

EMAIL VSALMASTLIAN@FENWICK.COM
Direct Dial +1 650-335-7853

David Stringer-Calvert
SRI International
333 Ravenswood Avenue
Menlo Park, CA 94025

Re: *IPA Technologies v. Amazon, et al.*; Case No. 16-cv-01266-RGA

Dear Mr. Stringer-Calvert:

Enclosed, please find a subpoena being served in connection with the above-captioned lawsuit, which is currently pending in the United States District Court for the District of Delaware. In this lawsuit, IPA Technologies is alleging that our clients, Amazon.com, Inc., and Amazon Digital Services, LLC infringe three United States patents.

The subpoena requests the production of specific documents identified in Exhibit A. We are, of course, willing to work with you to minimize the burden involved with your compliance with the subpoena. Please let me know if you have any questions or concerns.

Sincerely,

FENWICK & WEST LLP

/s/ *Vigen Salmastlian*

Vigen Salmastlian

VS:bjw
Enclosures

IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF DELAWARE

IPA TECHNOLOGIES, INC., Plaintiff, v. AMAZON.COM, INC., and AMAZON DIGITAL SERVICES LLC, Defendants.	The Honorable Richard G. Andrews Civil Action No. 1:16-CV-01266-RGA

NOTICE OF SUBPOENA TO SRI INTERNATIONAL

PLEASE TAKE NOTICE THAT, pursuant to Rule 45 of the Federal Rules of Civil Procedure, Defendants Amazon.com, Inc. and Amazon Digital Services LLC ("Amazon") is serving a subpoena upon SRI International requesting the production of documents, a copy of which is attached hereto.

Dated: July 17, 2019

Respectfully submitted,

ASHBY & GEDDES

/s/ Andrew C. Mayo

Steven J. Balick (#2114)
Andrew C. Mayo (#5207)
500 Delaware Avenue, 8th Floor
P.O. Box 1150
Wilmington, DE 19899
(302) 654-1888
sbalick@ashbygeddes.com
amayo@ashbygeddes.com

Attorneys for Defendants
AMAZON.COM, INC. and
AMAZON DIGITAL SERVICES LLC

Of Counsel:

J. David Hadden
Saina S. Shamilov
Ravi Ranganath
FENWICK & WEST LLP
801 California Street
Mountain View, CA 94041
(650) 988-8500

Todd R. Gregorian
Phillip J. Haack
Sapna S. Mehta
FENWICK & WEST LLP
555 California Street, 12th Floor
San Francisco, CA 94104
(415) 875-2300

Hilary L. Preston
VINSON & ELKINS LLP
666 Fifth Avenue, 26th Floor
New York, NY 10103-0040
(212) 237-0000

CERTIFICATE OF SERVICE

The undersigns certifies that on this 17th day of July, 2019, all counsel of record were served the foregoing by email.

ASHBY & GEDDES

/s/ Andrew C. Mayo

Steven J. Balick (#2114)
Andrew C. Mayo (#5207)
500 Delaware Avenue, 8th Floor
P.O. Box 1150
Wilmington, DE 19899
(302) 654-1888
sbalick@ashbygeddes.com
amayo@ashbygeddes.com

Attorneys for Defendants
AMAZON.COM, INC. and
AMAZON DIGITAL SERVICES LLC

AO 88B (Rev. 12/13) Subpoena to Produce Documents, Information, or Objects or to Permit Inspection of Premises in a Civil Action

UNITED STATES DISTRICT COURT

for the

District of Delaware

IPA Technologies, Inc.,

Plaintiff

v.

Amazon.com, Inc. and Amazon Digital Services LLC,

Defendant

Civil Action No. 1:16-CV-01266-RGA

SUBPOENA TO PRODUCE DOCUMENTS, INFORMATION, OR OBJECTS OR TO PERMIT INSPECTION OF PREMISES IN A CIVIL ACTION

To: SRI International

(Name of person to whom this subpoena is directed)

☑ *Production:* **YOU ARE COMMANDED** to produce at the time, date, and place set forth below the following documents, electronically stored information, or objects, and to permit inspection, copying, testing, or sampling of the material: See attached Exhibit A.

Place: Fenwick & West LLP 801 California Street, Mountain View, CA 94041 Telephone: (650) 988-8500	Date and Time: 08/01/2019 9:00 am

❒ *Inspection of Premises:* **YOU ARE COMMANDED** to permit entry onto the designated premises, land, or other property possessed or controlled by you at the time, date, and location set forth below, so that the requesting party may inspect, measure, survey, photograph, test, or sample the property or any designated object or operation on it.

Place:	Date and Time:

The following provisions of Fed. R. Civ. P. 45 are attached – Rule 45(c), relating to the place of compliance; Rule 45(d), relating to your protection as a person subject to a subpoena; and Rule 45(e) and (g), relating to your duty to respond to this subpoena and the potential consequences of not doing so.

Date: 07/17/2019

CLERK OF COURT

OR

Signature of Clerk or Deputy Clerk

/s/ Vigen Salmastlian

Attorney's signature

The name, address, e-mail address, and telephone number of the attorney representing *(name of party)* Defendants Amazon.com, Inc. and Amazon Digital Services LLC, who issues or requests this subpoena, are:

Vigen Salmastlian, Fenwick & West LLP, 801 California St., Mountain View, CA, tel. (650) 335-7853 vsalmastlian@fenwick.com

Notice to the person who issues or requests this subpoena

A notice and a copy of the subpoena must be served on each party in this case before it is served on the person to whom it is directed. Fed. R. Civ. P. 45(a)(4).

Civil Action No. 1:16-CV-01266-RGA

PROOF OF SERVICE

(This section should not be filed with the court unless required by Fed. R. Civ. P. 45.)

I received this subpoena for *(name of individual and title, if any)* ______________________

on *(date)* ____________ .

❐ I served the subpoena by delivering a copy to the named person as follows: ______________________

____ on *(date)* ____________ ; or

❐ I returned the subpoena unexecuted because: ______________________

____ .

Unless the subpoena was issued on behalf of the United States, or one of its officers or agents, I have also tendered to the witness the fees for one day's attendance, and the mileage allowed by law, in the amount of

$ ____________ .

My fees are $ ____________ for travel and $ ____________ for services, for a total of $ 0.00 .

I declare under penalty of perjury that this information is true.

Date: ____________

____
Server's signature

____
Printed name and title

____
Server's address

Additional information regarding attempted service, etc.:

Federal Rule of Civil Procedure 45 (c), (d), (e), and (g) (Effective 12/1/13)

(c) Place of Compliance.

(1) ***For a Trial, Hearing, or Deposition.*** A subpoena may command a person to attend a trial, hearing, or deposition only as follows:
(A) within 100 miles of where the person resides, is employed, or regularly transacts business in person; or
(B) within the state where the person resides, is employed, or regularly transacts business in person, if the person
(i) is a party or a party's officer; or
(ii) is commanded to attend a trial and would not incur substantial expense.

(2) ***For Other Discovery.*** A subpoena may command:
(A) production of documents, electronically stored information, or tangible things at a place within 100 miles of where the person resides, is employed, or regularly transacts business in person; and
(B) inspection of premises at the premises to be inspected.

(d) Protecting a Person Subject to a Subpoena; Enforcement.

(1) ***Avoiding Undue Burden or Expense; Sanctions.*** A party or attorney responsible for issuing and serving a subpoena must take reasonable steps to avoid imposing undue burden or expense on a person subject to the subpoena. The court for the district where compliance is required must enforce this duty and impose an appropriate sanction—which may include lost earnings and reasonable attorney's fees—on a party or attorney who fails to comply.

(2) ***Command to Produce Materials or Permit Inspection.***
(A) *Appearance Not Required.* A person commanded to produce documents, electronically stored information, or tangible things, or to permit the inspection of premises, need not appear in person at the place of production or inspection unless also commanded to appear for a deposition, hearing, or trial.
(B) *Objections.* A person commanded to produce documents or tangible things or to permit inspection may serve on the party or attorney designated in the subpoena a written objection to inspecting, copying, testing, or sampling any or all of the materials or to inspecting the premises—or to producing electronically stored information in the form or forms requested. The objection must be served before the earlier of the time specified for compliance or 14 days after the subpoena is served. If an objection is made, the following rules apply:
(i) At any time, on notice to the commanded person, the serving party may move the court for the district where compliance is required for an order compelling production or inspection.
(ii) These acts may be required only as directed in the order, and the order must protect a person who is neither a party nor a party's officer from significant expense resulting from compliance.

(3) ***Quashing or Modifying a Subpoena.***
(A) *When Required.* On timely motion, the court for the district where compliance is required must quash or modify a subpoena that:
(i) fails to allow a reasonable time to comply;
(ii) requires a person to comply beyond the geographical limits specified in Rule 45(c);
(iii) requires disclosure of privileged or other protected matter, if no exception or waiver applies; or
(iv) subjects a person to undue burden.
(B) *When Permitted.* To protect a person subject to or affected by a subpoena, the court for the district where compliance is required may, on motion, quash or modify the subpoena if it requires:
(i) disclosing a trade secret or other confidential research, development, or commercial information; or
(ii) disclosing an unretained expert's opinion or information that does not describe specific occurrences in dispute and results from the expert's study that was not requested by a party.
(C) *Specifying Conditions as an Alternative.* In the circumstances described in Rule 45(d)(3)(B), the court may, instead of quashing or modifying a subpoena, order appearance or production under specified conditions if the serving party:
(i) shows a substantial need for the testimony or material that cannot be otherwise met without undue hardship; and
(ii) ensures that the subpoenaed person will be reasonably compensated.

(e) Duties in Responding to a Subpoena.

(1) ***Producing Documents or Electronically Stored Information.*** These procedures apply to producing documents or electronically stored information:
(A) *Documents.* A person responding to a subpoena to produce documents must produce them as they are kept in the ordinary course of business or must organize and label them to correspond to the categories in the demand.
(B) *Form for Producing Electronically Stored Information Not Specified.* If a subpoena does not specify a form for producing electronically stored information, the person responding must produce it in a form or forms in which it is ordinarily maintained or in a reasonably usable form or forms.
(C) *Electronically Stored Information Produced in Only One Form.* The person responding need not produce the same electronically stored information in more than one form.
(D) *Inaccessible Electronically Stored Information.* The person responding need not provide discovery of electronically stored information from sources that the person identifies as not reasonably accessible because of undue burden or cost. On motion to compel discovery or for a protective order, the person responding must show that the information is not reasonably accessible because of undue burden or cost. If that showing is made, the court may nonetheless order discovery from such sources if the requesting party shows good cause, considering the limitations of Rule 26(b)(2)(C). The court may specify conditions for the discovery.

(2) ***Claiming Privilege or Protection.***
(A) *Information Withheld.* A person withholding subpoenaed information under a claim that it is privileged or subject to protection as trial-preparation material must:
(i) expressly make the claim; and
(ii) describe the nature of the withheld documents, communications, or tangible things in a manner that, without revealing information itself privileged or protected, will enable the parties to assess the claim.
(B) *Information Produced.* If information produced in response to a subpoena is subject to a claim of privilege or of protection as trial-preparation material, the person making the claim may notify any party that received the information of the claim and the basis for it. After being notified, a party must promptly return, sequester, or destroy the specified information and any copies it has; must not use or disclose the information until the claim is resolved; must take reasonable steps to retrieve the information if the party disclosed it before being notified; and may promptly present the information under seal to the court for the district where compliance is required for a determination of the claim. The person who produced the information must preserve the information until the claim is resolved.

(g) Contempt.
The court for the district where compliance is required—and also, after a motion is transferred, the issuing court—may hold in contempt a person who, having been served, fails without adequate excuse to obey the subpoena or an order related to it.

For access to subpoena materials, see Fed. R. Civ. P. 45(a) Committee Note (2013).

Exhibit A

EXHIBIT A

DEFINITIONS

1. The term "IPA" means IPA Technologies Inc., and includes any past and present parents (including, but not limited to Quarterhill, Inc. and WiLAN), predecessors, successors, subsidiaries, affiliates, divisions, associated organizations, joint ventures, and all present and former officers, directors, trustees, employees, staff members, agents, or other representatives, including counsel and patent agents, in any country.

2. The terms "you," "your," and "SRI" mean "SRI International" and includes any past and present parents, predecessors, successors, subsidiaries, affiliates, divisions, associated organizations, joint ventures, and all present and former officers, directors, trustees, employees, staff members, agents, or other representatives, including counsel and patent agents, in any country.

3. The term "Amazon" shall refer collectively to Amazon.com, Inc. and Amazon Digital Services, LLC.

4. The term "'115 patent" means U.S. Patent No. 6,851,115.

5. The term "'560 patent" means U.S. Patent No. 7,069,560.

6. The term "'128 patent" means U.S. Patent No. 7,036,128.

7. The term "Patents-In-Suit" shall refer, collectively, to the '115, '560, and '128 patents. Requests referring to "each of the Patents-In-Suit" (or "each asserted claim in the Patents-In-Suit") require responsive documents for each of the '115, '560, and '128 patents.

8. The term "related patents/applications" means any and all patents, patent applications and/or patent publications in the same patent family or families as the Patents-In-Suit and foreign counterparts; by way of example only, related patents/applications include any patent

document that (i) claims priority from any of the Patents-In-Suit, (ii) is identified as priority for any of the Patents-In-Suit, (iii) claims priority to any application to which any of the Patents-In-Suit claims priority, (iv) shares the same or substantially similar specification, or a portion thereof, as any of the Patents-in-Suit (including any provisional, divisional, continuation, and continuation-in-part applications), (v) is a reissue or reexamination of any of the Patents-in-Suit or any of the patents described in (i)-(iv), or is a foreign counterpart patent application or foreign counterpart patent to any of the patents or applications described in (i)-(iv).

9. The term "Related Litigation" refers to the consolidated action, *IPA Technologies Inc. v. Amazon.com, Inc.*, C.A. No. 1:16-cv-01266-RGA, pending in the United States District Court for the District of Delaware, and any other legal proceeding involving the Patents-In-Suit, any related patents, or any of IPA's predecessors-in-interest.

10. The term "prior art" means, as of the filing date of the Patents-In-Suit, any article, poster, abstract, chapter, display, slides, or other printed publication that discloses, or a use, sale, or offer for sale of a system or device disclosed or claimed in the Patents-In-Suit or that practices or could be used to practice, the alleged inventions or portions of the alleged inventions disclosed or claimed in the Patents-In-Suit or any other thing or activity which could be or could have been relied on by the United States Patent Office or a Court for an anticipation or obviousness determination of the Patents-In-Suit, including any and all patents, patent applications and/or publications prepared before the filing date of the Patents-In-Suit.

11. The term "SRI Product" means any SRI service, product, apparatus, device, technology, process, method, feature, component, software, hardware, firmware, application, source code, web page or pages, and any associated functionality, or any combination thereof.

12. The term "OAA" means all versions of the open agent architecture framework developed by SRI's Artificial Intelligence Center between January 1, 1993 and January 5, 1999, as mentioned, for example, in the OAA News, *available at* http://www.ai.sri.com/~oaa/news/news-v2.1.html (attached hereto as Ex. B), OAA 1.0 Documentation, *available at* http://www.ai.sri.com/~oaa/distribution/distribv1/documentation.html (attached hereto as Ex. C), and David L. Martin et. al., *The Open Agent Architecture: A Framework for Building Distributed Software Systems*, Artificial Intelligence Center SRI International (Oct. 19, 1998), *available at* http://www.ai.sri.com/~cheyer/papers/aai/oaa.html (attached hereto as Ex. D), including, but not limited to: software that enables coordination between facilitator agents and other agents; uses an agent registry that declares capabilities of software agents; performs speech recognition, natural language processing, data retrieval, data mining, and/or parsing of requests from users; implements the Interagent Communication Language (ICL); implements domain-specific, application-specific, and/or domain-independent strategies; and/or uses procedure solvables and data solvables that identify goals, parameters, and permissions.

13. The term "OAA Technology" means all versions of any SRI Product related to, incorporating, applying, using, or relying on OAA, including, but not limited to, (a) InfoWiz architecture and kiosk, (b) OAA-controlled robots (e.g., Saphira and Flakey), (c) Automated Office, (d) Unified Messaging, (e) CommandTalk, (f) Air Travel Information System (ATIS) and ATIS-Web, (g) Automatic Dialogue Summarizer (ADS), (h) MVIEWS, (i) Agent Development Tools, (j) InfoBroker, (k) Rental Agent, (l) DCG-NL agent, (m) GEN-NL agent, (n) Gemini Natural Language Understanding System, (o) Multi-Robot Control, (p) MARVEL, (q) SOLVIT, (r) Surgical Training, (s) Instant Collaboration, (t) Crisis Response, (u)

WebGrader, (v) DECIPHER, Multimodal Map, (w) MAESTRO, (x) SUO Robots, (y) CHeF, (z) TravelMate and CARS, and (aa) EMCE. *See, e.g.*, OAA-Based Applications *available at* http://www.ai.sri.com/~oaa/applications.html (attached hereto as Ex. E); David L. Martin et. al., *The Open Agent Architecture: A Framework for Building Distributed Software Systems*, Artificial Intelligence Center SRI International (Oct. 19, 1998), *available at* http://www.ai.sri.com/~cheyer/papers/aai/oaa.html (Ex. D); David Martin et. al. *Building and Using Practical Agent Applications*, SRI International PAAM '98 Tutorial, *available at* http://www.ai.sri.com/~oaa/distribution/doc/paam98-tutorial/sld001.htm (attached hereto as Ex. F).

14. The term "Interagent Communication Language" or "ICL" means the interface, communication, and task coordination language that is shared by software agents to perform queries, execute actions, exchange information, set triggers, and manipulate, read, and write data as mentioned, for example, in David L. Martin et. al., *The Open Agent Architecture: A Framework for Building Distributed Software Systems*, Artificial Intelligence Center SRI International (Oct. 19, 1998), *available at* http://www.ai.sri.com/~cheyer/papers/aai/oaa.html (Ex. D).

15. The term "ICL Technology" means technology related to ICL, including, but not limited to, Knowledge Query and Manipulation Language (KQML), Knowledge and Interchange Format (KIF), Multi-agent Environment for Constructing Cooperative Applications (MECCA), or any other related technology disclosed to or considered by the Foundation for Intelligent Physical Agents (FIPA). *See e.g.*, Adam Cheyer, "FIPA Report" (July 1, 1996), *available at* http://www.ai.sri.com/~oaa/distribution/MailArchive/oaa-users/0003.html (attached hereto as Ex. G).

16. The term "Named Inventor(s)" or "Inventors" shall mean one or more of the inventors named on any of the Patents-In-Suit, including Luc Julia, Adam Cheyer, and David L. Martin.

17. The terms "Covered Product" and "Covered Products" shall mean any SRI Product that SRI contends, or has ever contended, embodies or practices any of the alleged inventions disclosed or claimed in the Patents-In-Suit, and that was developed, sold, offered for sale, or made available to the public (whether or not currently sold, offered for sale, or available) by SRI or its licensees, assignees, or by any of their subsidiaries, affiliates, predecessors or successors-in-interest.

18. The term "document" has the broadest meaning accorded that term by Fed. R. Civ. P. 34(a) and includes, but is not limited to, all of the items defined in Fed. R. Evid. 1001, and all preliminary and final drafts of any such item.

19. The term "all documents" means any and all documents that you can locate through a diligent search of all locations likely to contain documents requested herein and through reasonable inquiry of all persons likely to know of the existence of documents requested herein. A draft or non-identical copy is a separate document within the meaning of this term. Any comment or notation appearing on any document, and not a part of the original text, is to be considered a separate "document."

20. The term "communication" shall refer to all written, oral, telephonic or other inquiries, dialogues, discussions, conversations, interviews, correspondence, consultations, negotiations, agreements, understandings, meetings, letters, notes, telegrams, advertisements, press releases, computer mail, e-mail and all other documents evidencing any verbal or nonverbal interaction between persons and/or entities.

21. The term "person" is defined as any natural person or any legal entity, including, without limitation, any business or governmental entity or association.

22. The terms "relate to," "relates to," "related to," "relating to," "referring to," "pertaining to," "pertain to," and "regarding" mean constitute, include, comprise, consist of, refer, reflect, discuss, show, state, explain, contradict, provide context to, evidence, concern or be in any way logically or factually connected with the matter discussed or identified.

23. The terms "or" and "and" shall be read in the conjunctive and in the disjunctive wherever they appear, and neither of these words shall be interpreted to limit the scope of these requests.

24. The term "any" and "each" should be understood to include and encompass "all."

25. All pronouns shall be construed to refer to the masculine, feminine, or neutral gender, in singular or plural, as in each case makes the request more inclusive.

26. The use of a verb in any tense shall be construed as including the use of the verb in all other tenses.

27. The singular form of any word shall be deemed to include the plural. The plural form of any word shall be deemed to include the singular.

INSTRUCTIONS

1. The subpoena served upon you commands you to produce all responsive documents in your possession, custody or control, including your attorneys, agents, representatives, or employees.

2. Pursuant to Federal Rule of Civil Procedure 45(d)(1), you are to provide documents responsive to this subpoena as they are kept in the ordinary course of business, or can organize and label the documents to correspond with the categories set forth below.

3. Electronic records and computerized information must be produced in an intelligible format, together with a description of the system from which they were derived sufficient to permit rendering the records and information intelligible.

4. Selection of documents from the files and other sources and the numbering of such documents shall be performed in such a manner as to ensure that the source of each document may be determined, if necessary.

5. File folders with tabs or labels or directories of files identifying documents called for by these requests must be produced intact with such documents.

6. Documents attached to each other shall not be separated.

7. Should you seek to withhold any document based on some limitation of discovery (including, but not limited to, a claim of privilege), supply a list of the documents for which limitation of discovery is claimed, indicating:

a. The identity of each document's author, writer, sender, or initiator;

b. The identity of each document's recipient, addressee, or person for whom it was intended;

c. The date of creation or transmittal indicated on each document, or an estimate of that date, indicated as such, if no date appears on the document;

d. The general subject matter as described on each document, or, if no such description appears, then some other description sufficient to identify the document; and

e. The claimed grounds for limitation of discovery (e.g., "attorney-client privilege").

8. If your response to a particular demand is a statement that you lack the ability to comply with that demand, you must specify whether the inability to comply is because the

particular item or category never existed, has been destroyed, has been lost, misplaced, or stolen, or has never been, or is no longer, in your possession, custody, or control, in which case the name and address of any person or entity known or believed by you to have possession, custody, or control of that document or category of document must be identified.

9. To the extent permitted and authorized by law, these document requests shall be deemed continuing and require supplemental responses and production if you obtain additional documents between the time of initial production and the time of hearing or trial.

DOCUMENT REQUESTS

1. All documents and things related to OAA, ICL, OAA Technology, and/or ICL Technology that was made, used, promoted, disclosed, publicly available, offered for sale, or sold before January 5, 1999, including, but not limited to, products, devices, components, guides, schematics, specifications, software, hardware, firmware, applications, source code, prototypes, physical devices, test plans, test results, demonstrations, videos, white papers, memoranda, theses, publications, presentations, meeting minutes, manuals, business plans, and marketing plans.

2. All documents and communications related to any version of OAA software, hardware, firmware, applications, source code, and prototypes that was promoted, disclosed, publicly tested, or publicly available before January 5, 1998, including, but not limited to, testing performed by "outside users" as identified at http://www.ai.sri.com/~oaa/distribution/MailArchive/oaa-users/0000.html (attached hereto as Ex. H) or members of any "OAA user group" as identified at http://www.ai.sri.com/~oaa/distribution/MailArchive/oaa-users/0001.html (attached hereto as Ex. I), and all recipients of and responses to each communication identified at http://www.ai.sri.com/~oaa/distribution/MailArchive/oaa-users/ (attached hereto as Ex. J).

3. All versions of "OAA 1.0 Documentation" identified at http://www.ai.sri.com/~oaa/distribution/distribv1/documentation.html (Ex. C) that was promoted, disclosed, or publicly available before January 5, 1998, including, but not limited to "Frequently Asked Questions," "OAA 1.0 Tutorial," "OAA 1.0 Developer's Guide," "Agent Development Tools (ADT) User's Guide," "OAA Reference Manual," "OAA-USERS mailing list archive," "OAA 1 Lisp Library Overview and Developer's Notes," and "OAA Agent Documentation."

4. All versions of software, hardware, firmware, applications, source code, prototypes, and physical devices used to demonstrate Automated Office and Multimodal Maps, including, for example, the demonstrations available at http://www.ai.sri.com/~oaa/distribution/doc/videos/office.avi and http://www.ai.sri.com/movies/sfmap.avi.

5. All documents and things related to the "tutorial on constructing distributed agent applications . . . presented by David Martin and Adam Cheyer at the 3rd International conference on The Practical Application of Intelligent Agent and Multi-Agents, Monday March 23-Wednesday March 25, 1998, London, UK" identified at http://www.ai.sri.com/~oaa/news/news-v2.1.html (Ex. B), including, but not limited to, all versions of the presentation submitted prior to the March 25, 1998 conference and related communications.

6. All documents and things related to the conception, research, design, development, testing, reduction to practice, manufacture, disclosure, promotional activity, use, offer for sale, and sale of the subject-matter of the Patents-In-Suit or any Covered Product.

7. Documents sufficient to identify each project that each Named Inventor worked on while employed by SRI before January 5, 1999.

8. Documents sufficient to identify each individual's involvement in and contribution to the conception, research, design, development, testing, reduction to practice, or manufacture of

the subject-matter of the Patents-In-Suit.

9. All documents related to each invention disclosure identifying any of the subject-matter of the Patents-in-Suit, the decision to file and/or prosecute each patent application disclosing any of the subject-matter of the Patents-in-Suit, and each internal and/or external person involved in the prosecution of such patent applications.

10. All documents and things related to invalidity of any claim of the Patents-in-Suit or any prior art identified by SRI before or at the time of conception or development of any of the subject-matter of the Patents-In-Suit, including, but not limited to, any prior art in the same field as any Covered Products, OAA, OAA Technology, ICL, and/or ICL Technology.

11. All documents and things related to the Patents-In-Suit or any related patents/applications submitted to and received from the U.S. Patent and Trademark Office and similar agencies of other countries.

12. All documents relating to any legal proceeding involving, OAA, ICL, OAA Technology, ICL Technology, and/or Covered Products.

13. All documents and communications related to any assignment of the Patents-In-Suit to and from SRI.

14. All license agreements related to the Patents-In-Suit and related patents/applications and all related communications.

15. All documents and communications related to SRI's assignment and/or license of any patents to Siri, Inc., including, but not limited to, the amount Siri, Inc. paid for each of the Patents-in-Suit, any related patents/applications, and any other SRI patents.

16. All documents and communications related to SRI's assignment and/or license of any patents to IPA, including, but not limited to, any consideration provided by IPA for each of

the Patents-in-Suit, any related patents/applications, and any other SRI patents.

17. All agreements related to SRI's employment of the Named Inventors and SRI's spin out of Siri, Inc.

18. All documents and communications related to any efforts to license, assign, sell, or provide a right to use the Patents-In-Suit (both individually or collectively) or any related patents/applications regardless of whether such efforts were successful.

19. All documents related to the sale of any Covered Product, OAA Technology, or any SRI Product incorporating, applying, using, or relying on OAA, ICL, or ICL Technology, whether sold by SRI or any third party, including, but not limited, gross sales and revenue on a year-by-year and product-by-product basis.

20. All documents and things related to the publication entitled "MVIEWS: Multimodal Tools for the Video Analyst" presented at the 1998 international conference on Intelligent User Interfaces in San Francisco, CA between January 6-9, 1998 (attached hereto as Ex. K), including, but not limited to, each disclosure or submission of the publication before January 5, 1998 to conference organizers, conference participants, publishers, entities offering printing services, or any other third party.

21. All documents and things related to the disclosure, promotion, consideration, evaluation, or adoption of OAA, ICL, OAA Technology, ICL Technology, or any prior art to the Patents-in-Suit by FIPA or any third-party attending any FIPA meeting before January 5, 1998, including, but not limited to, the FIPA meeting in New York attended by Adam Cheyer identified at http://www.ai.sri.com/~oaa/distribution/MailArchive/oaa-users/0003.html (Ex. G).

Exhibit B

The OAA[TM] News

Previous Issue

Vol 2, no. 1, January 1, 1998

OAA 2.0 Tutoral at PAAM-98!

A tutorial on constructing distributed agent applications will be be presented by David Martin and Adam Cheyer at the 3rd International conference on The Practical Application of Intelligent Agents and Multi-Agents, Monday March 23 - Wednesday March 25, 1998, London, UK. This tutorial, aimed at software developers, describes several different types of agent-based systems and strategies for implementing them. Examples and lessons learned will be drawn from the application areas that SRI has been involved in over the last few years, including

- Personal assistants
- Agent-based support for emergency response teams
- Intelligence analysis environments
- Agents for information search, retrieval, and management
- Coordination of robotic agents

Subject to time constraints, the presentation will include:

- Systems integration
    - Agentification of legacy systems
    - Interoperability across heterogeneous languages and platforms
    - Integrating with object-oriented standards
- Tools & techniques
    - Design of agent-based systems
    - Tools for specifying individual agents and communities of agents
    - Reuse of agents
    - Debugging distributed systems
- User Interfaces to distributed agents
    - Multimodal interfaces
    - Ubiquitous access to agent services
    - Collaboration among multiple humans/agents

- Animated personas
- Agent-based implementation of interfaces

The tutorial will also include an introduction to programming within SRI's Open Agent Architecture™, the agent-based framework used to implement the applications mentioned above. A public version of OAA™ 2.0 will soon be available for download from the Web.

The presenters, from the AI Center of SRI International, have been involved in the development of agent technologies since 1993.

Exhibit C

OAA 1.0 Documentation

Warning! Although some of this documentation is incomplete or out of date, current work is focusing on OAA 2.0 and it's documentation, so this will probably stay somewhat out of date...

OAA Documentation

- Frequently Asked Questions (FAQ)
- OAA 1.0 Tutorial
- OAA 1 Developer's Guide
- Agent Development Tools User Manual
- OAA Reference Manual -- Delphi edition
- OAA Reference Manual -- Java edition
- OAA 1 Lisp Library Overview and Developer's Notes
- OAA-Users mailing list archive
- OAA Agent Documentation
- OAA Presentation

OAA Frequently Asked Questions (FAQ)

The FAQ contains general questions and answers about the Open Agent Architecture, including a section on Troubleshooting. The FAQ is available only in HTML form.

OAA 1.0 Tutorial

The OAA 1.0 Tutorial is a brief, step-by-step guide for new developers for running an OAA system and adding a new agent. The tutorial is currently available only in HTML form.

OAA 1.0 Developer's Guide

This developer's guide contains information on:

- Overview of the OAA
- Agent Infrastructure
- Requesting and providing services using the Interagent Communication Language (ICL)
- Create OAA Agents
- OAA Agent Library Library primitives
- Triggers
- Data Management
- Running an agent application using Start-It

This documentation can be accessed in HTML form, PostScript, and in Japanese (HTML). For the Japanese version, in Netscape, set Language Encoding option to "Japanese (Autoselect)"

Agent Development Tools (ADT) Users's Guide

This user's manual describes in detail how to use the ADT to create new OAA agents and projects. Unfortunately, ADT cannot be made available for use outside SRI. But it is documented here, and in a journal paper (Agent development tools for the open agent architecture), for those who may be interested in its concepts.

This documentation can be accessed in HTML form, or in PostScript.

OAA Reference Manual -- Delphi version

This manual gives reference pages on all of OAA's built-in primitives supplied by the Agent Library. This version of the reference manual is specifically for the Delphi Programming language, but it should be fairly useful for other languages as well. This documentation can be accessed in HTML form.

OAA Reference Manual -- Java version

This manual gives reference pages on all of OAA's built-in primitives supplied by the Agent Library. This version of the reference manual is specifically for the Java Programming language, but it should be fairly useful for other languages as well. (Generated using javadoc) This documentation can be accessed in HTML form.

OAA-USERS mailing list archive

Discussions among OAA users are archived here. You may access the archive:

- By Subject
- By Author
- By Thread
- By Date

OAA 1 Lisp Library Overview and Developer's Notes

This documentation can be accessed in HTML form.

OAA Agent Documentation

You may access online documentation about individual OAA agents by either browsing or by using one of the searches below.

Browse

- OAA Agent documentation

Exhibit D

The Open Agent Architecture:
A Framework for Building Distributed Software Systems

David L. Martin
Adam J. Cheyer
Douglas B. Moran
Artificial Intelligence Center
SRI International
333 Ravenswood Avenue
Menlo Park, CA 94025 USA
+1 650 859 5552
{martin,cheyer,moran}@ai.sri.com

Abbreviated title: *Open Agent Architecture*

Abstract

The Open Agent Architecture (OAA), developed and used for several years at SRI International, makes it possible for software services to be provided through the cooperative efforts of distributed collections of autonomous agents. Communication and cooperation between agents are brokered by one or more facilitators, which are responsible for matching requests, from users and agents, with descriptions of the capabilities of other agents. Thus, it is not generally required that a user or agent know the identities, locations, or number of other agents involved in satisfying a request. OAA is structured so as to minimize the effort involved in creating new agents and "wrapping" legacy applications, written in various languages and operating on various platforms; to encourage the reuse of existing agents; and to allow for dynamism and flexibility in the makeup of agent communities. Distinguishing features of OAA as compared with related work include extreme flexibility in using facilitator-based delegation of complex goals, triggers, and data management requests; agent-based provision of multimodal user interfaces; and built-in support for including the user as a privileged member of the agent community.

This paper explains the structure and elements of agent-based systems constructed using OAA. The characteristics and use of each major component of OAA infrastructure are described, including the agent library, the Interagent Communication Language, capabilities declarations, service requests, facilitation, management of data repositories, and autonomous monitoring using triggers. To provide technical context, we describe the motivations for OAA's design, and situate its features within the realm of alternative software paradigms. A summary is given of OAA-based systems built to date, and brief descriptions are given of several of these.

1 Introduction

The evolution of models for the design and construction of software systems is being driven forward by several closely interrelated trends:

- The adoption of a *networked computing model* is leading to a greatly increased reliance on distributed sites for both data and processing resources. Indeed, with a reported 1800 new computers being added to the Internet every day, a paradigm shift for computing is well under way, one which moves away from requiring all relevant data and programs to reside on the user's desktop machine. The data now routinely accessed from computers spread around the world has become increasingly rich in format, comprising multimedia documents, and audio and video streams; with the popularization of JAVA, it may also include programs that can be downloaded and executed on the local machine. As we become increasingly reliant on networked computing, we need approaches to software design that allow for flexible composition of distributed processing elements in a dynamically changing and relatively unstable environment.

- In an increasing variety of domains, application designers and users are coming to expect the deployment of *smarter, longer-lived, more autonomous, software applications.* Push technology, persistent monitoring of information sources, and the maintenance of user models, allowing for personalized responses and sharing of preferences, are examples of the simplest manifestations of this trend. Commercial enterprises are introducing significantly more advanced approaches, in many cases employing recent research results from artificial intelligence, data mining, machine learning, and other fields.

- More than ever before, the increasing complexity of systems, the development of new technologies, and the availability of multimedia material and environments are creating a demand for *more accessible, more intuitive user interfaces.* Autonomous, distributed, multicomponent systems providing sophisticated services will no longer lend themselves to the familiar "direct manipulation" model of interaction, in which an individual user masters a fixed selection of commands provided by a single application. Ubiquitous computing, in networked environments, has brought about a situation in which the typical user of many software services is likely to be a nonexpert, who may access a given service infrequently or only a few times. Accommodating such usage patterns calls for new approaches. Fortunately, input modalities now becoming widely available, such as speech recognition and pen-based handwriting/gesture recognition, and the ability to manage the presentation of systems' responses by using multiple media provide an opportunity to fashion a style of human-computer interaction that draws much more heavily on our experience with human-human interactions.

The Open Agent Architecture Architecture (OAA), [1] a framework for constructing multiagent systems developed at the Artificial Intelligence Center of SRI International, arose from a

[1] Open Agent Architecture and OAA are trademarks of SRI International. Other brand names and product names herein are trademarks and registered trademarks of their respective holders.

desire to accommodate developments in these three areas in an integrated framework, which is suitable for practical use. In Sections 2 and 3 of this paper, we first review various approaches to distributed computing, and then situate our own approach within the scope of this related work. Following that, we briefly characterize the range of OAA-based systems built to date. Subsequent sections provide detailed descriptions of the inner workings of OAA. Whereas the motivating concepts for an early version of OAA were presented in (Cohen et al., 1994), and certain OAA-based systems have been described in (Cheyer and Julia, 1995; Martin et al., 1996; Martin et al., 1997; Moore et al., 1996; Moran et al., 1997; Moran and Cheyer, 1995), this is the first paper to present a detailed technical explanation of the system-building resources provided by OAA.

2 Technologies for Distributed Computing

We briefly review the overall concepts, advantages, and disadvantages of several relevant approaches to distributed computing, including distributed objects, mobile objects, blackboard-style architectures, and agent-based software engineering.

2.1 The Distributed Object Approach

Object-oriented languages, such as C++ or JAVA, provide significant advances over standard procedural languages with respect to the reusability and modularity of code:

- Encapsulation: encourages the creation of library interfaces that minimize dependencies on underlying algorithms or data structures. Changes to programming internals can be made at a later date with requiring modifications to the code that uses the library.
- Inheritance: permits the extension and modification of a library of routines and data without requiring source code to the original library.
- Polymorphism: allows one body of code to work on an arbitrary number of data types.

Whereas "standard" object-oriented programming (OOP) languages can be used to build monolithic programs out of many object building blocks, distributed object technologies (DOOP) such as OMG's CORBA ((OMG), 1997) or Microsoft's DCOM (Microsoft, 1996) allow the creation of programs whose components may be spread across multiple machines. To implement a client-server relationship between objects, distributed object systems use a registry mechanism (CORBA's registry is called an Object Request Broker, or ORB) to store the interface descriptions of available objects. Through the ORB's services, a client can transparently invoke a method on a remote server object; the ORB is responsible for finding an object that can implement the request, passing it the parameters, invoking its method, and returning the results. The client does not have to be aware of where the object

is located, its programming language, its operating system, or any other system aspects that are not part of an object's interface.

Although distributed objects offer a powerful paradigm for creating networked applications, certain aspects of the approach are not perfectly tailored to the constantly changing environment of the Internet. A major restriction of the DOOP approach is that the interactions among objects are fixed through explicitly coded instructions by the application developer. This implies that it is very difficult to reuse an object in a new application without bringing along all its inherent dependencies on other objects (embedded interface definitions and explicit method calls). Another restriction of the DOOP approach is the result of its reliance on a remote procedure call (RPC) style of communication. Although easy to debug, this single thread of execution model does not facilitate programming to exploit the potential for parallel computation that one would expect in a distributed environment. In addition, RPC uses a blocking (synchronous) scheme that does not scale well for high-volume transactions.

2.2 Mobile Objects

Mobile objects, sometimes called mobile agents, are bits of code that can move to another execution site (presumably on a different machine) under their own programmatic control, where they can then efficiently interact with the local environment. Commercial instantiations of this technology include Aglets from IBM, Concordia from Mitsubishi, and Voyager from ObjectSpace.

For certain types of problems, the mobile object paradigm offers advantages over more traditional distributed object approaches. These advantages include

- Network bandwidth: for some database queries or electronic commerce applications, it is more efficient to perform tests on data by bringing the tests to the data than by bringing large amounts of data to the testing program.
- Parallelism: mobile agents can be spawned in parallel to accomplish many tasks at once.

Disadvantages (or inconveniences) of the mobile agent approach are that

- In a fashion similar to that of DOOP programming, an agent developer must programmatically specify where to go and how to interact with the target environment.
- There is generally little coordination support to encourage interactions among multiple (mobile) participants.
- Agents must be written in the programming language supported by the execution environment, whereas many other distributed technologies support heterogeneous communities of components, written in diverse programming languages.

2.3 Blackboard Architectures

Blackboard approaches, such as Schwartz's FLiPSiDE (Schwartz, 1995) or Gelernter's LINDA (Gelernter, 1993), allow multiple processes to communicate by reading and writing tuples from a global data store. Each process can watch for items of interest, perform computations based on the state of the blackboard, and then add partial results or queries that other processes can consider.

Blackboard architectures provide a flexible framework for problem solving by a dynamic community of distributed processes. A blackboard approach provides one solution to eliminating the tightly bound interaction links that some of the other distributed technologies require during interprocess communication. This advantage can also be a disadvantage: although a programmer does not need to refer to a specific process during computation, the framework does not provide programmatic control for doing so in cases where this would be practical.

2.4 Agent-based Software Engineering

Several research communities have approached distributed computing by casting it as a problem of modeling communication and cooperation among autonomous entities. Effective communication among independent actors requires four components: (1) a transport mechanism carrying messages in an asynchronous fashion, (2) an interaction protocol defining various types of communication interchange and their social implications (for instance, a response is expected of a question), (3) a content language permitting the expression and interpretation of utterances, and (4) an agreed-upon set of shared vocabulary and meaning for concepts (often called an *ontology*). Such mechanisms permit a much richer style of interaction among participants than can be expressed using a distributed object's RPC model or a blackboard architecture's centralized exchange approach.

Undoubtably, the most widely used foundation technology for agent-based software engineering is the Knowledge Query and Manipulation Language (KQML) (Labrou and Finin, 1997; Finin et al., 1997). KQML, which specifies an interaction protocol, is often used in conjunction with the Knowledge Interchange Format (KIF) (Genesereth and Fikes, 1992) as content language, and either ad hoc or more formalized ontologies. KQML introduced the use of symbolic *performatives* to capture information about the purpose of a communication, and its place within a conversation. Although creating a standardized representation for conversational interactions is one important aspect of multiagent cooperation, KQML is limited by its reliance on a fixed core set of atomic performatives, and the inevitable difficulty in arriving at just the right set capable of expressing every kind interaction and service request.

Another influential approach, which makes stronger assumptions about the knowledge and processing used within individual agents, is based on the structuring of the agents' activities around the concepts of Belief, Desire, and Intention (BDI) (Rao and Georgeff, 1995). While BDI's emphasis on a higher level of abstraction has been extremely important in giving direction to work on agent-based systems, its applicability may be limited by the structural requirements imposed on individual agents, and by difficulties in interoperating with legacy

systems.

3 Philosophy and Goals of OAA

Our approach to distributed computing shares much in common with the paradigms outlined above. As with distributed object frameworks, the primary goal of OAA is to provide a means for integrating heterogeneous applications in a distributed infrastructure. However, we have also sought to incorporate some of the dynamism and extensibility of blackboard approaches, the efficiency associated with mobile objects, and the rich and complex interactions of communicating agents. Here, we spell out in greater detail the goals of OAA, which may be categorized under the general headings of *interoperation and cooperation, user interfaces*, and *software engineering.*

Versatile mechanisms of interoperation and cooperation. *Interoperation* refers to the ability of distributed software components – agents – to communicate meaningfully. While every system-building framework must provide mechanisms of interoperation at some level of granularity, agent-based frameworks face important new challenges in this area. This is true primarily because autonomy, the hallmark of *individual* agents, necessitates greater flexibility in interactions within *communities* of agents. *Coordination* refers to the mechanisms by which a community of agents is able to work together productively on some task. In these areas, the goals for our framework are to

- *Provide flexibility in assembling communities of autonomous service providers* — both at development time and at runtime. Agents that conform to the linguistic and ontological requirements for effective communication should be able to participate in an agent community, in various combinations, with minimal prerequisite knowledge of the characteristics of the other players. Agents with duplicate and overlapping capabilities should be able to coexist within the same community, with the system making the best possible use of the redundancy.

- *Provide flexibility in structuring cooperative interactions* among the members of a community of agents. A framework should provide economical means of setting up a variety of interaction patterns among agents, without requiring an inordinate amount of complexity or infrastructure within the individual agents. The provision of a service should not be dependent upon a particular configuration of agents.

- *Impose the right amount of structure* on individual agents. Different approaches to the construction of multiagent systems impose different requirements on the individual agents. For example, because KQML is neutral as to the content of messages, it imposes minimal structural requirements on individual agents. On the other hand, the BDI paradigm is likely to impose much more demanding requirements, because it makes assumptions about the nature of the programming elements that are meaningful to individual agents. OAA falls somewhere between the two; our goal has been to provide a rich set of interoperation and coordination, but without precluding any of the software engineering goals defined below.

- *Include legacy and "owned-elsewhere" applications.* Whereas *legacy* usually implies reuse of an established system fully controlled by the agent-based system developer, *owned-elsewhere* refers to applications to which the developer has partial access, but no control. Examples of the latter are data sources and services available on the World Wide Web, via simple form-based interfaces, and applications used cooperatively within a virtual enterprise, which remain the property of separate corporate entities. It must be possible for both classes of application to interoperate, more or less as full-fledged members of the agent community, without requiring an overwhelming integration effort.

Human-oriented user interfaces. Systems composed of multiple distributed components, and possibly dynamic configurations of components, require the crafting of intuitive user interfaces to

- Provide *conceptually natural* means of interacting with multiple distributed components. When there are numerous disparate agents, and/or complex tasks implemented by the system, the user should be able to express requests without having detailed knowledge of the individual agents. With speech recognition, handwriting recognition, and natural language technologies becoming more mature, an agent architecture must be prepared for these forms of input to play an increased role in the tasking of agent communities.

- Treat *users as privileged members* of the agent community. By providing an appropriate level of task specification within *software* agents, and reusable means of translating between this level and the level of *human* requests, it should be possible to construct interactions that seamlessly incorporate both types of "agent".

- Support *collaboration* (simultaneous work over shared data and processing resources) between users and agents.

Realistic software engineering requirements. To be successful, a system-building framework must address the practical concerns of real-world applications, as expressed by these goals:

- *Minimize the effort* required to create new agents, and to wrap existing applications.

- *Encourage reuse,* both of domain-independent and domain-specific components. The concept of *agent orientation*, like that of object orientation, provides a natural conceptual framework for reuse, so long as mechanisms for encapsulation and interaction are structured appropriately.

- *Support lightweight, mobile platforms.* Such platforms should be able to serve as hosts for agents, without requiring the installation of a massive environment. It should also be possible to construct individual agents that are relatively small and modest in their processing requirements.

- *Minimize platform and language barriers.* Creation of new agents, as well as wrapping of existing applications, should not require the adoption of a new language or environment.

4 Overview of OAA

In this section, we present an overview of OAA, first describing the basic components and structure of the framework, and then illustrating these concepts with a sample application.

4.1 OAA System Structure

Figure 1 presents the structure typical of a small OAA system, showing a user interface agent and several application agents and meta-agents, organized as a community of peers by their common relationship to a facilitator agent.

The facilitator is a specialized server agent that is responsible for coordinating agent communications and cooperative problem-solving. In many systems, the facilitator is also used to provide a global data store for its client agents, which allows them to adopt a blackboard style of interaction. Note that a system configuration is not limited to a single facilitator. Larger systems can be assembled from multiple facilitator/client groups, each having the sort of structure shown in Figure 1.

The other categories of agents illustrated here – application agents, meta-agents, and user interface agents – are categories recognized by convention only; that is, they are not formally distinguished within the system. Application agents are usually specialists that provide a collection of services of a particular sort. These services could be domain-independent technologies (such as speech recognition, natural language processing, email, and some forms of data retrieval and data mining) or user-specific or domain-specific (such as a travel planning and reservations agent). Application agents may be based on legacy applications or libraries, in which case the agent may be little more than a wrapper that calls a pre-existing API.

Meta-agents are those whose role is to assist the facilitator agent in coordinating the activities of other agents. While the facilitator possesses domain-independent coordination strategies, meta-agents can augment these by using domain- and application-specific knowledge or reasoning (rules, learning algorithms, planning, and so forth).

The user interface agent plays an extremely important and interesting role in many OAA systems. In some systems, this agent is implemented as a collection of "micro-agents", each monitoring a different input modality (point-and-click, handwriting, pen gestures, speech), and collaborating to produce the best interpretation of the current inputs. These micro-agents are shown in Figure 1 as Modality Agents.

All agents that are *not* facilitators are referred to as *client* agents — so called because each acts (in some respects) as a client of some facilitator, which provides communication and other essential services for the client. When invoked, a client agent makes a connection to

a facilitator, which is known as its *parent facilitator*. Upon connection, an agent informs its parent facilitator of the services it can provide. When the agent is needed, the facilitator sends it a request expressed in the Interagent Communication Language (ICL). The agent parses this request, processes it, and returns answers or status reports to the facilitator. In processing a request, the agent can make use of a variety of capabilities provided by OAA. For example, it can use ICL to request services of other agents, set triggers, and read or write shared data on the facilitator (or other client agents that maintain shared data).

The common infrastructure for constructing agents is supplied by an *agent library*, which is available in several different programming languages. The library has been designed to minimize the effort required to construct a new system, and to maximize the ease with which legacy systems can be agentified.

4.2 Sample Interactions

Perhaps the best way to obtain an intuitive sense of how the OAA typically functions is to briefly look at an example of how OAA has been applied to a real application. In the Automated Office system, a mobile executive with a telephone and a laptop computer can access and task commercial applications such as calendars, databases, and email systems running back at the office. As depicted in Figure 2, an application agent provides a wrapper for each program, making its functionality and natural language vocabulary available to the agent community through registration with a facilitator.

A user interface (UI) agent, shown in Figure 3, runs on the user's local laptop and is responsible for accepting user input, sending requests to the facilitator for delegation to appropriate agents, and displaying the results of the distributed computation. The user may interact directly with a specific remote application by clicking on active areas in the interface, calling up a form or window for that application, and making queries with standard interface dialog mechanisms. Conversely, a user may express a task to be executed by using typed, handwritten, or spoken (over the telephone) English sentences, without explicitly specifying which agent or agents should perform the task. For instance, if the question "What is my schedule?" is written in the user interface, this request will be sent by the UI to the facilitator, which in turn will ask a natural language (NL) agent to translate the query into ICL. To accomplish this task, the NL agent may itself need to make requests of the agent community to resolve unknown words such as "me" (the UI agent can respond with the name of the current user) or "schedule" (the calendar agent defines this word). The resulting ICL expression is then routed by the facilitator to appropriate agents (in this case, the calendar agent) to execute the request. Results are sent back to the UI agent for display.

The spoken request "When mail arrives for me about security, notify me immediately." produces a slightly more complex example involving communication among all agents in the system. After translation into ICL as described above, the facilitator installs a trigger on the mail agent to look for new messages about security. When one such message does arrive in its mail spool, the trigger fires, and the facilitator matches the action part of the trigger to capabilities published by the notification agent. The notification agent is an

example of a meta-agent, as it makes use of rules concerning the optimal use of different output modalities (email, fax, speech generation over the telephone) plus information about an individual user's preferences to determine the best way of relaying a message through available media transfer application agents. After some competitive parallelism to locate the user (the calendar and database agents may have different guesses as to where to find the user) and some cooperative parallelism to produce required information (telephone number of location, user password, and an audio file containing a text-to-speech representation of the email message), a telephone agent can call the user, verify identity through touchtones, and then play the message.

Some key ideas illustrated by the above examples are the following:

1. As new agents connect to the facilitator, registering capability specifications and natural language vocabulary, what the user can say and do dynamically changes.

2. The interpretation and execution of a task is a distributed process, with no one agent defining the set of possible inputs to the system.

3. A single request can produce cooperation and flexible communication among many agents, written in different programming languages and spread across multiple machines.

In our following detailed view of the Open Agent Architecture, we order the presentation top-down, beginning with the means by which a group of agents works together, then considering the mechanisms that support the use of shared data repositories and triggers, and finally describing some of the basic infrastructure underlying the construction of individual agents. To illustrate the technical aspects of the approach, we describe several applications implemented within the OAA.

5 Mechanisms of Cooperation

Cooperation among the agents of an OAA system is achieved via messages expressed in a common language, ICL, and is normally structured around a three-part approach: providers of services register capabilities specifications with a facilitator, requesters of services construct goals and relay them to a facilitator, and facilitators coordinate the efforts of the appropriate service providers in satisfying these goals.

5.1 The Interagent Communication Language

OAA's Interagent Communication Language (ICL) is the interface, communication, and task coordination language shared by all agents, regardless of what platform they run on or what computer language they are programmed in. ICL is used by an agent to task itself or some

subset of the agent community, either using explicit control or, more frequently, in an under-specified, loosely constrained manner. OAA agents employ ICL to perform queries, execute actions, exchange information, set triggers, and manipulate data in the agent community.

One of the fundamental program elements expressed in ICL is the *event*. The activities of every agent, as well as communications between agents, are structured around the transmission and handling of events. In communications, events serve as messages between agents; in regulating the activities of individual agents, they may be thought of as goals to be satisfied.

Each event has a type, a set of parameters, and content. For example, the agent library procedure *oaa_Solve* can be used by an agent to request services of other agents. A call to *oaa_Solve*, within the code of agent A, results in an event having the form

ev_post_solve(Goal, Params)

going from A to the facilitator, where *ev_post_solve* is the type, *Goal* is the content, and *Params* is a list of parameters. The allowable content and parameters vary according to the type of the event.

The ICL includes a layer of conversational protocol, similar in spirit to that provided by KQML, and a content layer, analogous to that provided by KIF. The conversational layer of ICL is defined by the event types, together with the parameter lists associated with certain of these event types. The content layer consists of the specific goals, triggers, and data elements that may be embedded within various events.

The conversational protocol is specified using an orthogonal, parameterized approach. That is, the conversational aspects of each element of an interagent conversation are represented by a selection of an event type, in combination with a selection of values for an orthogonal set of parameters. This approach offers greater expressiveness than an approach based solely on a fixed selection of *speech acts*, such as embodied in KQML. For example, in KQML, a request to satisfy a query can employ either of the performatives *ask_all* or *ask_one*. In ICL, on the other hand, this type of request is expressed by the event type *ev_post_solve*, together with the *solution_limit(N)* parameter – where N can be any positive integer. (A request for all solutions is indicated by the omission of the *solution_limit* parameter.) The request can also be accompanied by other parameters, which combine to further refine its semantics.

In KQML, then, this example forces one to choose between two possible conversational options, neither of which may be precisely what is desired. In either case, the performative chosen is a single value that must capture the entire conversational characterization of the communication. This requirement raises a difficult challenge for the language designer, to select a set of performatives that provides the desired functionality without becoming unmanageably large. Consequently, the debate over the right set of performatives has consumed much discussion within the KQML community.

The content layer of the ICL has been designed as an extension of the PROLOG programming language, to take advantage of unification and other features of PROLOG. OAA's agent libraries (especially the non-PROLOG versions) provide support for constructing, parsing, and manipulating ICL expressions.

While it is possible to embed content expressed in other languages within an ICL event, it is advantageous to express content in ICL wherever possible. The primary reason for this is to allow the facilitator access to the content, as well as the conversational layer, in delegating requests. Not only does this give the facilitator more information about the nature of a request, but it also makes it possible for the facilitator to decompose compound requests, and individually delegate the subrequests.

Important declarations and other program elements represented using ICL expressions include, in addition to events, capabilities declarations, requests for services, responses to requests, trigger specifications, and shared data elements.

5.2 Providing Services

Every agent participating in an OAA-based system defines and publishes a set of capabilities declarations, expressed in ICL, describing the services that it provides. These declarations establish a high-level interface to the agent. This interface is used by a facilitator in communicating with the agent, and, most important, in delegating service requests (or parts of requests) to the agent. Partly due to the use of PROLOG as the basis of ICL, we refer to these capabilities declarations as *solvables*.

Two major types of solvables are distinguished: *procedure* solvables and *data* solvables. Intuitively, a procedure solvable performs a test or action, whereas a data solvable provides access to a collection of data. For example, in creating an agent for a mail system, procedure solvables might be defined for sending a message to a person, testing whether a message about a particular subject has arrived in the mail queue, or displaying a particular message onscreen. For a database wrapper agent, one might define a distinct data solvable corresponding to each of the relations present in the database. Often, a data solvable is used to provide a *shared* data store, which may be not only queried, but also updated, by various agents having the required permissions.

Technically, the primary differences between the two types of solvables are these: First, each procedure solvable must have a handler declared and defined for it, whereas this is not necessary for a data solvable. (The handling of requests for a data solvable is provided transparently by the agent library.) Second, data solvables are associated with a dynamic collection of facts (or clauses), which may be modified at runtime, both by the agent providing the solvable, and by other agents (provided they have the required permissions). Third, special features, available for use with data solvables, facilitate maintaining the associated facts. Some of these features are mentioned in Section 6.

In spite of these differences, it should be noted that the means of *use* (that is, the means by which an agent requests a service) is the same for the two types of solvables. Requesting of services is described in Section 5.3.

A request for one of an agent's services normally arrives in the form of an event from the agent's facilitator. The appropriate handler then deals with this event. The handler may be coded in whatever fashion is most appropriate, depending on the nature of the task, and

the availability of task-specific libraries or legacy code, if any. The only hard requirement is that the handler return an appropriate response to the request, expressed in ICL. Depending on the nature of the request, this response could be an indication of success or failure, or a list of solutions (when the request is a data query).

The agent library provides a set of procedures allowing an agent to add, remove, and modify its solvables, which it may do at any time after connecting to its facilitator.

5.2.1 Specification of Solvables

A solvable has three parts: a *goal*, a list of *permissions*, and a list of *parameters*, which are declared using the format

```
solvable(Goal, Parameters, Permissions)
```

The goal of a solvable, which syntactically takes the form of an ICL structure, is a logical representation of the service provided by the solvable. (An ICL structure consists of a *functor* with 0 or more arguments. For example, in the structure a(b,c), 'a' is the functor, and 'b' and 'c' the arguments.) As with a PROLOG structure, the goal's arguments may themselves be structures.

Various options can be included in the parameters list, to refine the semantics associated with the solvable. First and foremost, the *type* parameter is used to say whether the solvable is *data* or *procedure*. When the type is *procedure*, another parameter may be used to indicate the handler to be associated with the solvable. Some of the parameters appropriate for a *data* solvable are mentioned in Section 6.

In either case (procedure or data solvable), the *private* parameter may be used to restrict the use of a solvable to the declaring agent. This parameter is valuable when the agent intends the solvable to be solely for its internal use and wants to take advantage of OAA mechanisms in accessing it, or when the agent wants the solvable to be available to outside agents only at selected times. In support of the latter case, it is possible for the agent to change the status of a solvable from private to nonprivate at any time.

The permissions of a solvable provide the means by which an agent may control access to its services. They allow the agent to restrict calling and writing of a solvable to itself and/or other selected agents. (*Calling* means requesting the service encapsulated by a solvable, whereas *writing* means modifying the collection of facts associated with a data solvable.) The default is for every solvable to be callable by anyone, and for data solvables to be writable by anyone. A solvable's permissions can be changed at any time, by the agent providing the solvable.

For example, the solvables of a simple email agent might include

```
solvable(send_message(email, +ToPerson, +Params),
         [type(procedure), callback(send_mail)],
         [])
```

```
solvable(last_message(email, -MessageId),
         [type(data), single_value(true)],
         [write(true)]),
solvable(get_message(email, +MessageId, -Msg),
         [type(procedure), callback(get_mail)],
         [])
```

The symbols '+' and '-', indicating input and output arguments, are at present used only for purposes of documentation. Most parameters and permissions have default values, and specifications of default values may be omitted from the parameters and permissions lists.

A programmer who defines an agent's capabilities in terms of solvable declarations is, in a sense, creating the vocabulary with which other agents will communicate with the new agent. The problem of ensuring that agents will speak the same language and share a common, unambiguous semantics of the vocabulary, is called the *ontology problem*. The OAA provides a few tools (see more about agent development tools in (Martin et al., 1996)) and services (automatic translations of solvables by the facilitator) to help minimize this issue; however, the OAA still must rely on vocabulary from either formally engineered ontologies for specific domains (for instance, see http://www-ksl.stanford.edu/knowledge-sharing/ontologies/html/) or on ontologies constructed during the incremental development of a body of agents for several applications.

Although OAA imposes no hard restrictions (other than the basic syntax) on the form of solvable declarations, two common usage conventions illustrate some of the utility associated with solvables.

- Classes of services are often tagged by a particular type. For instance, in the example above, the "last_message" and "get_message" solvables are specialized for email, not by modifying the *names* of the services, but rather by the use of the 'email' parameter, which serves during the execution of an ICLrequest to select (or not) a specific type of message.
- Actions are generally written using an imperative verb as the functor of the solvable, the direct object (or item class) as the first argument of the predicate, required arguments following, and then an extensible parameter list as the last argument. The parameter list can hold optional information usable by the function. The ICLexpression generated by a natural language parser often makes use of this parameter list to store prepositional phrases and adjectives.

As an illustration of the above two points, "Send mail to Bob about lunch" will be translated into an ICLrequest send_message(email, 'Bob Jones', [subject(lunch)]), whereas "Remind Bob about lunch" would leave the transport unspecified (send_message(KIND, 'Bob Jones', [subject(lunch)])), enabling all available message transfer agents (e.g., fax, phone, mail, pager) to compete for the opportunity to carry out the request.

5.3 Requesting Services

An agent requests services of the community by delegating tasks or goals to its facilitator. Each request contains calls to one or more agent solvables, and optionally specifies parameters containing advice to help the facilitator determine how to execute the task. It is important to note that calling a solvable does *not* require that the agent specify (or even know of) a particular agent or agents to handle the call. While it is possible to specify one or more agents using an address parameter (and there are situations in which this is desirable), in general it is advantageous to leave this delegation to the facilitator. Programming in this style greatly reduces the hard-coded dependencies among components that one often finds in other distributed frameworks.

The OAA libraries provide an agent with a single, unified point of entry for requesting services of other agents: the library procedure *oaa_Solve*. In the style of logic programming, *oaa_Solve* may be used both to retrieve data and to initiate actions. To put this another way, calling a *data* solvable looks the same as calling a *procedure* solvable.

5.3.1 Compound Goals

One of the most powerful features of OAA is the ability of a client agent (or a user) to submit compound goals to a facilitator. A compound goal is composed using operators similar to those employed by PROLOG, that is, the comma for conjunction, the semicolon for disjunction, and the arrow for conditional execution. Three of the several significant extensions to PROLOG syntax and semantics are of particular interest here. First, a "parallel disjunction" operator indicates that the disjuncts are to be executed (by different agents) simultaneously. Second, it is possible to specify whether a given subgoal is to be executed breadth-first or depth-first. [2] Third, each subgoal of a compound goal can have an address and/or a set of parameters attached to it. Thus, each subgoal takes the form

Address:Goal::Parameters

where both *Address* and *Parameters* are optional.

An address, if present, specifies one or more agents to handle the given goal, and may employ several different types of referring expression: unique names, symbolic names, and shorthand names. Every agent has a unique name, assigned by its facilitator, which relies upon network addressing schemes to ensure its global uniqueness. Agents also have self-selected symbolic names (for example, "mail"), which are not guaranteed to be unique. When an address includes a symbolic name, the facilitator takes this to mean that all agents having that name should be called upon. Shorthand names include 'self' and 'parent' (which refers to the agent's facilitator). We emphasize that the address associated with a goal or subgoal is always optional. When an address is not present, it is the facilitator's job to supply an appropriate address, as explained in Section 5.5.

[2] This capability is under development.

The distributed execution of compound goals becomes particularly powerful when used in conjunction with natural language or speech-enabled interfaces, as the query itself may specify how functionality from distinct agents will be combined. As a simple example, the spoken utterance "Fax it to Bill Smith's manager." can be translated into the following compound ICL request:

```
oaa_Solve((manager('Bill Smith', M), fax(it,M,[])), [strategy(action)])
```

5.4 Refining Service Requests

The parameters associated with a goal (or subgoal) can draw on useful features to refine the request's meaning. For example, it is frequently important to be able to specify whether or not solutions are to be returned synchronously; this is done using the *reply* parameter, which can take any of the values *synchronous, asynchronous*, or *none*. As another example, when the goal is a noncompound query of a data solvable, the *cache* parameter may be used to request local caching of the facts associated with that solvable. Many of the remaining parameters fall into two categories: advice and feedback.

Feedback parameters allow a service requester to receive information from the facilitator about how a goal was handled. This feedback can include such things as the identities of the agents involved in satisfying the goal, and the amount of time expended in the satisfaction of the goal.

Advice parameters give constraints or guidance for the facilitator to use in completing and interpreting the goal. For example, the *solution_limit* parameter allows the requester to say how many solutions it is interested in; the facilitator and/or service providers are free to use this information in optimizing their efforts. Similarly, *time_limit* is used to say how long the requester is willing to wait for solutions to its request, and, in a multifacilitator system, *level_limit* may be used to say how remote the facilitators may be that are consulted in the search for solutions. The *priority* parameter is used to indicate that a request is more urgent than previous requests that have not yet been satisfied. Other advice parameters are used to tell the facilitator whether parallel satisfaction of the parts of a goal is appropriate, how to combine and filter results arriving from multiple solver agents, and whether the requester itself may be considered a candidate solver of the subgoals of a request.

As mentioned in section 5.1, advice parameters are intended to provide an extensible set of low-level, orthogonal parameters capable of combining with the ICL goal language to fully express how information should flow among participants. Multiple parameters can be grouped together and given a group name; the resulting *high-level advice parameters* can be used to express concepts analogous to KQML's performatives, but also to define classifications of problem types. For instance, KQML's "ask_all" and "ask_one" performatives would be represented as combinations of values given to the parameters *reply*, *parallel_ok*, and *solution_limit*. As an example of a higher-level problem type, the strategy "math_problem" might send the query to all appropriate math solvers in parallel, collect their responses, and signal a conflict if different answers are returned. The strategy "essay_question" would send

the request to all appropriate participants, and signal a problem (i.e., cheating) if any of the returned answers are identical.

When a facilitator receives a compound goal, its job is to construct a goal satisfaction plan and oversee its satisfaction in the most appropriate, efficient manner that is consistent with the specified advice.

5.5 Facilitation

Facilitation plays a central role in OAA. At its core, our notion of facilitation is similar to that proposed by Genesereth (Genesereth and Singh, 1993) and others. In short, a facilitator maintains a knowledge base that records the capabilities of a collection of agents, and uses that knowledge to assist requesters and providers of services in making contact. But our notion of facilitation is also considerably stronger in four respects.

First, it encompasses a very general notion of *transparent delegation*, which means that a requesting agent can generate a request, and a facilitator can manage the satisfaction of that request, without the requester needing to have any knowledge of the identities or locations of the satisfying agents. In some cases, such as when the request is a data query, the requesting agent may also be oblivious to the *number* of agents involved in satisfying a request. Transparent delegation is possible because agents' capabilities (solvables) are treated as an abstract description of a service, rather than as an entry point into a library or body of code.

Second, an OAA facilitator is distinguished by its handling of compound goals (introduced in Section 5.3.1). This involves three types of processing: *delegation*, that is, determination of who (which specific agents) will execute a compound goal and how (combination and routing of results from subgoals); *optimization* of the completed goal, including parallelization where appropriate; and *interpretation* of the optimized goal. The *delegation* step results in a goal that is unambiguous as to its meaning and as to the agents that will participate in satisfying it. Completing the addressing of a goal involves the selection of one or more agents to handle each of its subgoals (that is, each subgoal for which this selection has not been specified by the requester). In doing this, the facilitator uses its knowledge of the capabilities of its client agents (and possibly of other facilitators, in a multifacilitator system). It may also use strategies or advice specified by the requester, as explained below. The *optimization* step results in a goal whose interpretation will require as few exchanges as possible, between the facilitator and the satisfying agents, and can exploit parallel efforts of the satisfying agents, wherever this does not affect the goal's meaning. The *interpretation* of a goal involves the coordination of requests to the satisfying agents, and assembling their responses into a coherent whole, for return to the requester.

The third respect in which OAA facilitation extends the basic concept of facilitation is that the facilitator can employ strategies and advice given by the requesting agent, thus resulting in a variety of interaction patterns that may be instantiated in the satisfaction of a request. Some of these strategies are mentioned in Section 5.4, and additional possibilities under consideration are mentioned in Section 11.

Finally, the OAA concept of facilitation has been generalized so as to handle the distribution of both data update requests and requests for installation of triggers, using some of the same strategies that are employed in the delegation of service requests. (Triggers and data maintenance mechanisms are discussed in sections 7 and 6 respectively.)

It should be noted that the reliance on facilitation is not absolute; that is, there is no hard requirement that requests and services be matched up by the facilitator, or that interagent communications go through the facilitator. (Indeed, as mentioned elsewhere, there is support in the agent library for explicit addressing of requests, and planned support for peer-to-peer communications.) However, OAA has been designed so as to encourage developers to employ the paradigm of community, and to minimize their development effort in doing so, by taking advantage of the facilitator's provision of transparent delegation and handling of compound goals.

In summary, we stress that a facilitator is always viewed as a *coordinator*, not a controller, of cooperative task completion. The facilitator never initiates an activity, but rather responds to requests to manage the satisfaction of some goal, the update of some data repository, or the installation of a trigger by the appropriate agent or agents. This approach makes it possible for all agents to take advantage of the facilitator's expertise in delegation, and its up-to-date knowledge about the current membership of a dynamic community. In addition, in many situations, the facilitator's coordination services allows the developer to lessen the complexity of individual agents, resulting in a more manageable software development process, and enabling the creation of lightweight agents.

6 Maintaining Data Repositories

The agent library supports the creation, maintenance, and use of databases, in the form of data solvables. Creation of a data solvable requires only that it be declared, as explained in Section 5.2.1. Querying a data solvable, as with access to any solvable, is done using *oaa_Solve*. Here, we clarify the ways in which these solvables are maintained and used, and mention some of the features associated with them.

A data solvable is conceptually the same as a relation in a relational database. The facts associated with each solvable are maintained by the agent library, which also handles incoming messages containing queries of data solvables. It is possible to refine the default behavior of the library in managing these facts, using parameters specified with the solvable's declaration. For example, the parameter *single_value* is used to indicate that the solvable should only contain a single fact at any given point in time. The parameter *unique_values* indicates that no duplicate values should be stored.

Other parameters can allow data solvables to make use of the concepts of ownership and persistence. Because data solvables are often used to implement shared repositories, it can be useful to maintain a record of which agent created each fact of a solvable; this agent is considered to be the fact's owner. In many applications, it is useful to have an agent's facts removed when that agent goes offline (that is, the agent is no longer participating in

the agent community, whether by deliberate termination or by malfunction). When a data solvable is declared to be nonpersistent, its facts are automatically maintained in this way, whereas a persistent data solvable retains its facts until they are explicitly removed.

The agent library provides procedures by which agents can update (add, remove, and replace) facts belonging to data solvables, either locally or on other agents, given that they have the required permissions. These procedures may be refined using many of the same parameters that apply to service requests. For example, the *address* parameter is used to specify one or more particular agents to which the update request applies. In its absence, just as with service requests, the update request goes to *all* agents providing the relevant data solvable. This default behavior can be used to maintain coordinated "mirror" copies of a data set within multiple agents, and can be useful in support of distributed, collaborative activities.

Similarly, the *feedback* parameters, described in connection with *oaa_Solve*, are also available for use with data maintenance requests.

The ability to provide data solvables is not limited to client agents; data solvables can also be maintained by a facilitator, at the request of a client of the facilitator, and their maintenance and use shared by all the facilitator's clients. This can be a useful strategy with a relatively stable collection of agents, where the facilitator's workload is predictable.

6.1 Using a Blackboard Style of Communication

When a data solvable is publicly readable and writable, it may be thought of as a global data repository, which can be used cooperatively by a group of agents. In combination with the use of triggers, this allows the agents to organize their efforts around a "blackboard" style of communication.

As an example, the "DCG-NL" agent (one of several existing natural language processing agents), which provides natural language processing services for a variety of its peer agents, expects those other agents to record, on the facilitator, the vocabulary to which they are prepared to respond, with an indication of each word's part of speech, and of the logical form (ICL subgoal) that should result from the use of that word. To make this possible, when it comes online, the NL agent installs a data solvable for each basic part of speech on its facilitator. For instance, one such solvable would be

```
solvable(noun(Meaning, Syntax), [], [])
```

(Note that the empty lists for the solvable's permissions and parameters are acceptable here, since the default permissions and parameters provide appropriate functionality.)

In the Office Assistant system, several agents make use of these services. For instance, the database agent uses the following call, to library procedure *oaa_AddData*, to post the noun 'boss', and to indicate that the "meaning" of boss is the concept 'manager':

```
oaa_AddData(noun(manager, atom(boss)), [address(parent)])
```

7 Autonomous Monitoring with Triggers

OAA triggers provide a general mechanism for requesting that some action be taken when some set of conditions is met. Each agent can install triggers either locally, for itself, or remotely, on its facilitator or peer agents. There are four types of triggers: communication, data, task, and time. In addition to a type, each trigger specifies a condition and an action, both expressed in ICL. The condition indicates under what circumstances the trigger should fire, and the action indicates what should happen when it fires. In addition, each trigger can be set to fire either an unlimited number of times, or a specified number of times, which can be any positive integer.

Triggers are used in a wide variety of ways within OAA systems, for example, for monitoring external sensors in the execution environment, tracking the progress of complex tasks, or coordinating communications between agents that are essential for the synchronization of related tasks. The installation of a trigger within an agent can be thought of as a representation of that agent's *commitment* to carry out the specified action, whenever the specified condition holds true.

The four types of triggers can be characterized informally as follows:

- *Communication triggers* allow any incoming or outgoing event (message) to be monitored. For instance, a simple communication trigger may say something like

  "Whenever a solution to a goal is returned from the facilitator, send the result to the presentation manager to be displayed to the user."

- *Data triggers* monitor the state of a data repository (which can be maintained on a facilitator or a client agent). Data triggers' conditions may be tested upon the addition, removal, or replacement of a fact belonging to a data solvable. An example data trigger is

  "When 15 users are simultaneously logged on to a machine, send an alert message to the system administrator."

- *Task triggers* contain arbitrary conditions that are tested after the processing of each incoming event and whenever a timeout occurs in the event polling. These conditions may specify any goal executable by the local ICL interpreter, and most often are used to test when some solvable becomes satisfiable.

  Task triggers are useful in checking for task-specific internal conditions. Although in many cases such conditions are captured by solvables, in other cases they may not be. For example, a mail agent might watch for new incoming mail, or an airline database agent may monitor which flights will arrive later than scheduled. An example task trigger is

  "When mail arrives for me about security, notify me immediately."

- *Time triggers* monitor time conditions. For instance, an alarm trigger can be set to fire at a single fixed point in time (e.g., "On December 23rd at 3pm"), or on a recurring basis (e.g., "Every three minutes from now until noon").

Triggers are implemented as data solvables, declared implicitly for every agent. When requesting that a trigger be installed, an agent may use many of the same parameters that apply to service and data maintenance requests.

One important feature of OAA triggers is that, in contrast with most programming methodologies, the agent on which the trigger is installed only has to know how to evaluate the conditional part of the trigger, not the consequence – when the trigger fires, the action is delegated to the facilitator for execution. Whereas many commercial mail programs allow rules of the form "When mail arrives about XXX, [forward it, delete it, archive it]", the possible actions are hard-coded and the user must select from a fixed set. In OAA, the consequence may be any compound goal executable by the dynamic community of agents. Since new agents define both functionality and vocabulary, when an unanticipated agent (for example, a fax agent) joins the community, no modifications to existing code is required for a user to make use of it – "When mail arrives, fax it to Bill Smith."

8 The Agent Library

OAA's agent library, which provides the necessary infrastructure for constructing an agent-based system, is available in several programming languages, including PROLOG, C, C++, JAVA, LISP, VISUAL BASIC, and DELPHI. As mentioned earlier, two goals of the library's design have been to minimize the effort required to construct a new system, and to maximize the ease with which legacy systems can be agentified.

The library's several families of procedures, provide all the functionalities mentioned in this paper, as well as many that are omitted, for lack of space. For example, declarations of an agent's solvables, and their registration with a facilitator, are managed using procedures such as *oaa_Declare*, *oaa_Undeclare*, and *oaa_Redeclare*. Updates to data solvables can be accomplished with a family of procedures including *oaa_AddData*, *oaa_RemoveData*, and *oaa_ReplaceData*. Similarly, triggers are maintained using procedures such as *oaa_AddTrigger*, *oaa_RemoveTrigger*, and *oaa_ReplaceTrigger*.

The essential elements of protocol (that is, the details of the messages that encapsulate a service request and its response) are provided by the library, and made transparent in so far as possible, so that application code can be simpler. This enables the developer to focus on the desired functionality, rather than on the details of message construction and communication. For example, to request a service of another agent, an agent calls the library procedure *oaa_Solve*. This call results in a message to a facilitator, which will exchange messages with one or more service providers, and then send a message containing the desired results to the requesting agent. These results are returned via one of the arguments of *oaa_Solve*. None of the messages involved in this scenario is explicitly constructed by the agent developer. (Note that this is a description of the *synchronous* use of *oaa_Solve*.)

The agent library provides both *intra*agent and *inter*agent infrastructure; that is, mechanisms supporting the internal structure of individual agents, on the one hand, and mechanisms of cooperative interoperation between agents, on the other. It is worth noting that most of

the infrastructure cuts across this boundary; that is, many of the same mechanisms support both agent internals and agent interactions in an integrated fashion. For example, services provided by an agent can be accessed by that agent through the same procedure (*oaa_Solve*) that it would employ to request a service of another agent (the only difference being in the *address* parameter accompanying the request). This, in turn, helps the developer to reuse code and avoid redundant entry points into the same functionality.

Both of the characteristics described above (transparent construction of messages and integration of *intra*agent with *inter*agent mechanisms) apply to most other library functionality as well, including data management and temporal control mechanisms.

9 OAA Applications

The OAA has been used to implement more than fifteen applications integrating such diverse technologies as image processing, speech recognition, multiuser collaboration, text extraction, planning, and virtual reality. Table 1 is a partial list of OAA-based applications from which we will take a few examples to illustrate qualities important to the framework.

9.1 Unified Messaging

The Unified Messaging application extends the Automated Office presented in section 4 with an emphasis on ubiquitous access and dynamic presentation of the information and services supported by the agent community. The agents used in this application are depicted in Figure 4.

A real dialog taken from the current system can provide insight into how systems are built using OAA. In this scenario, the user, with only a telephone as an interface, is planning a trip to Boston where he will soon give a presentation. Capitalized sentences are phrases spoken by the user into the telephone and processed by a phone agent. Responses, unless otherwise indicated, are spoken by the system using text-to-speech generation.

```
1.1  Welcome to SRI International.  Please enter your user ID and password.
     <Enters touchtone ID and password>
     Good to see you again Adam Cheyer.  I am listening to you.
```

Every user interface agent, including the telephone agent, should know the identify of its user. This information is used in resolving anaphoric references such as “me” and “I”, and allows multiple user interfaces operated by the same user to work together.

```
1.2  WHAT IS TODAY'S SCHEDULE?
     Here is today's schedule for Adam Cheyer:
       At 10am for 1 hour, meeting with Dave.
```

```
  At 3pm for 1 hour, presentation about software agents.
End of schedule.
```

If the user is operating both a graphical user interface and a telephone, as described in conjunction with the Automated Office application, the result of this spoken request is to display a calendar window containing the current schedule. In this case, with no graphical display available, the GEN_NL agent is tasked to produce a spoken response that can be played over the phone. GEN_NL shares the same dynamic vocabulary and phrasal rules as the natural language parser DCG_NL, and contains strategies for producing responses to queries using either simple or list-based multimedia utterances.

```
1.3  FIND FRIDAY'S WEATHER IN BOSTON.
     The weather in Boston for Friday is as follows:
       Sunny in the morning. Partly cloudy in the afternoon with a 20
       percent chance of thunderstorms late. Highs in the mid 70s.
```

In addition to data accessible from legacy applications, content may be retrieved by web-reading agents which provide OAA wrappers around useful websites.

```
1.4  FIND ALL NEW MAIL MESSAGES.
     There are 2 messages available.
     Message 1, from Mark Tierny, entitled ``OAA meeting.''

1.5  NEXT MESSAGE
     Message 2, from Jennifer Schwefler, entitled ``Presentation Summary.''

1.6  PLAY IT.
     This message is a multipart MIME-encoded message.  There are two parts.
     Part 1.  (Voicemail message, not text-to speech):
     Thanks for taking part as a speaker in our conference.
     The schedule will be posted soon on our homepage.

1.7  NEXT PART
     Part 2. (read using text-to-speech):
     The presentation home page is http://www....

1.8  PRINT MESSAGE
     Command executed.
```

Mail messages are no longer just simple text documents, but often consist of multiple subparts containing audio files, pictures, webpages, attachments and so forth. When a user asks to play a complex email message over the telephone, many different agents may be implicated in the translation process, which would be quite different given the request "print it." The challenge is to develop a system which will enable agents to cooperate in an extensible, flexible manner that alleviates explicit coding of agent interactions for every possible input/output combination.

In an OAA implementation, each agent concentrates only on what it can do and on what it knows, and leaves other work to be delegated to the agent community. For instance, a printer agent, defining the solvable print(Object,Parameters), can be defined by the following pseudocode, which basically says, "If someone can get me a document, in either POSTSCRIPT or text form, I can print it.".

```
print(Object, Parameters) {

   ' If Object is reference to ``it'', find an appropriate document
   if (Object = "ref(it)")
      oaa_Solve(resolve_reference(the, document, Params, Object),[]);

   ' Given a reference to some document, ask for the document in POSTSCRIPT
   if (Object = "id(Pointer)")
      oaa_Solve(resolve_id_as(id(Pointer), postscript, [], Object),[]);

   ' If Object is of type text or POSTSCRIPT, we can print it.
   if ((Object is of type Text) or (Object is of type Postscript))
      do_print(Object);
}
```

In our example, since an email message is the salient document, the mail agent will receive a request to produce the message as POSTSCRIPT. Whereas the mail agent may know how to save a text message as POSTSCRIPT, it will not know what to do with a webpage or voicemail message. For these parts of the message, it will simply send oaa_Solve requests to see if another agent knows how to accomplish the task.

Until now, the user has been using only a telephone as user interface. Now, he moves to his desktop, starts a web browser, and accesses the URL referenced by the mail message.

```
1.9  RECORD MESSAGE
     Recording voice message.  Start speaking now.

1.10 THIS IS THE UPDATED WEB PAGE CONTAINING THE PRESENTATION SCHEDULE.
     Message one recorded.

1.11 IF THIS WEB PAGE CHANGES, GET IT TO ME WITH NOTE ONE.
     Trigger added as requested.
```

In this example, a local agent which interfaces with the web browser can return the current page as a solution to the request "oaa_Solve(resolve_reference(this, web_page, [], Ref),[])", sent by the NL agent. A trigger is installed on a web agent to monitor changes to the page, and when the page is updated, the notify agent can find the user and transmit the webpage and voicemail message using the most appropriate media transfer mechanism.

This example based on the Unified Messaging application is intended to show how OAA concepts can be used to produce a simple yet extensible solution to a multiagent problem

that would be difficult to implement using a more rigid framework. The application supports adaptable presentation for queries across dynamically changing, complex information; shared context and reference resolution among applications; and flexible translation of multimedia data. In the next section, we will present an application which highlights the use of parallel competition and cooperation among agents during multimodal fusion.

9.2 Multimodal Map

The goal of the Multimodal Map application is to explore natural ways of communicating with a community of agents. Inspired by the way a professor would instruct his students at a blackboard, through combinations of drawing, writing, speaking, gesturing, circling, underlining and so forth, the Multimodal Map provides an interactive interface on which the user may draw, write or speak. In a travel planning domain (Figure 5), available information includes hotel, restaurant, and tourist-site data retrieved by distributed software agents from commercial Internet sites. The types of user interactions and multimodal issues handled by the application can be illustrated by a brief scenario from (Cheyer et al., 1998) featuring working examples taken from the current system.

```
Sara is planning a business trip to San Francisco, but would like to
schedule some activities for the weekend while she is there.  She turns
on her laptop PC, executes a map application, and selects San Francisco.

2.1    [Speaking] Where is downtown?
       Map scrolls to appropriate area.
2.2    [Speaking and drawing region] Show me all hotels near here.
       Icons representing hotels appear.
2.3    [Writes on a hotel] Info?
       A textual description (price, attributes, etc.) appears.
2.4    [Speaking] I only want hotels with a pool.
       Some hotels disappear.
2.5    [Draws a crossout on a hotel that is too close to a highway]
       Hotel disappears
2.6    [Speaking and circling] Show me a photo of this hotel.
       Photo appears.
2.7    [Points to another hotel]
       Photo appears.
2.8    [Speaking] Price of the other hotel?
       Price appears for previous hotel.
2.9    [Speaking and drawing an arrow] Scroll down.
       Display adjusted.
2.10   [Speaking and drawing an arrow toward a hotel]
       What is the distance from this hotel to Fisherman's Wharf?
       Distance displayed.
2.11   [Pointing to another place and speaking] And the distance to here?
```

```
      Distance displayed.

Sara decides she could use some human advice.  She picks up the phone,
calls Bob, her travel agent, and writes Start collaboration to synchronize
his display with hers. At this point, both are presented with identical
maps, and the input and actions of one will be remotely seen by the other.

3.1   [Sara speaks and circles two hotels]
      Bob, I'm trying to choose between these two hotels.  Any opinions?
3.2   [Bob draws an arrow, speaks, and points]
      Well, this area is really nice to visit. You can walk there from
      this hotel.
      Map scrolls to indicated area.  Hotel selected.
3.3   [Sara speaks] Do you think I should visit Alcatraz?
3.4   [Bob speaks] Map, show video of Alcatraz.
      Video appears.
3.5   [Bob speaks] Yes, Alcatraz is a lot of fun.
```

For this system, the main research focus is on how to generate the most appropriate interpretation for the incoming streams of multimodal input. Besides providing a user interface *to* a dynamic set of distributed agents, the application is built *using* an agent framework, with the OAA helping coordinate competition and cooperation among information sources, which work in parallel to resolve the ambiguities arising at every level of the interpretation process:

- Low-level processing of the data stream: Pen input may be interpreted as a gesture (e.g., 2.5: crossout) by one algorithm, or as handwriting by a separate recognition process (e.g., 2.3: "info?"). Multiple hypotheses may be returned by a modality recognition component.

- Anaphora resolution: When resolving anaphoric references, separate information sources may contribute to resolving the reference:

  - Context by object type: For an utterance such as "show photo of the hotel", the natural language component can return a list of the last hotels talked about.

  - Deictic: In combination with a spoken utterance like "show photo of this hotel", pointing, circling, or arrow gestures might indicate the desired object (e.g., 2.7). Deictic references may occur before, during, or after an accompanying verbal command.

  - Visual context: Given the request "display photo of the hotel", the user interface agent might determine that only one hotel is currently visible on the map, and therefore this might be the desired reference object.

  - Database queries: Information from a database agent can be combined with results from other resolution strategies. Examples are "show me a photo of the hotel in Menlo Park" and 2.2.
  - Discourse analysis: Discourse can provide a source of information for phrases such as "No, the other one" (or 2.8).

  This list is by no means exhaustive. Examples of other resolution methods include spatial reasoning ("the hotel between Fisherman's Wharf and Lombard Street") and user preferences ("near my favorite restaurant").

- Cross-modality influences: When multiple modalities are used together, one modality may reinforce or disambiguate the interpretation of another. For instance, the interpretation of an arrow gesture may vary when accompanied by different verbal commands (e.g., "scroll left" vs. "show info about this hotel"). In the latter example, the system must take into account how accurately and unambiguously an arrow selects a single hotel.

- Addressee: With the addition of collaboration technology, humans and automated agents all share the same workspace. A pen doodle or a spoken utterance may be meant for either another human, the system (3.1), or both (3.2).

The implementation of the Multimodal Map application exploits several features of the OAA:

- Reference resolution and task delegation are handled in a distributed fashion by the parallel parameters of oaa_Solve, with meta-agents encoding rules to help the facilitator make context- or user-specific decisions about priorities among knowledge sources.

- Basic multiuser collaboration is handled through OAA's built-in data management services. The map user interface publishes data solvables for elements such as icons, screen position, and viewers, and defines these elements to have the attribute "shareable". For every update to this public data, the changes are automatically replicated to all members of the collaborative session, with associated callbacks producing the visible effect of the data change (e.g., adding or removing an icon).

- Functionality for recording and playback of a session is easily implemented by adding agents as members of the collaborative community. These agents either record the data changes to disk, or read a logfile and replicate the changes in the shared environment.

- The domain-specific code for interpreting travel planning dialog is cleanly separated from the speech, natural language, pen recognition, database and map user interface agents. These components were reused without modification to add multimodal map capabilities to other applications for activities such as crisis management, multi-robot control, and the MVIEWS tools for the video analyst.

10 Related Work

Agent-based systems have shown much promise for flexible, fault-tolerant, distributed problem solving. Much of the foundational work on agent technology has focused on interagent communication protocols (Finin et al., 1997), patterns of conversation for agent interactions (FIPA, 1997), and basic facilitation capabilities, including agent name servers and other types of registry services (*e.g.*, brokers, matchmakers) (Sycara et al., 1996).

Because there is insufficient space here to cover the gamut of work on agent architectures, we restrict ourselves to mentioning several projects that have helped to evolve some notion of facilitation. Genesereth has emphasized the role of a facilitator (Genesereth and Singh, 1993; Genesereth and Katchpel, 1994), and in (Genesereth and Singh, 1993) describes a facilitator based on logical reasoning. This facilitator shares our emphasis on content-based routing and the synthesis of complex multistep delegation plans, but does not go as far as OAA in allowing the service requester to influence the strategies used by the facilitator. Similarly, the InfoSleuth system (Nodine and Unruh, 1997) employs matchmaking agents having the ability to reason deductively about whether expressions of requirements (by requesters) match with the advertised capabilities of service providers. KQML (Labrou and Finin, 1997; Finin et al., 1997) provides "capability-definition performatives", such as *advertise*, and "facilitation performatives", such as *broker_one* and *broker_all*. While these performatives may be suitable for structuring the basic interactions between the players in a facilitated system, it should be noted that they provide only a communication protocol. That is, the specific strategies employed by a facilitator, and the means of advising a facilitator in selecting a strategy, are beyond the scope of KQML specifications. Sycara et al. delineate the concepts of matchmaking, brokering, and facilitation in a useful way, and explore the tradeoffs inherent in the use of these approaches. Overall, they find that a brokered or facilitated system can exhibit dramatically better performance than one based on matchmaking.

11 Future Directions

Much work remains to be done, both at implementation and conceptual levels. Areas for further investigation include scalability, robustness (fault tolerance), improved development and runtime tools, and improved facilitation strategies and services.

The use of facilitators offers both advantages and weaknesses with respect to scalability and fault tolerance. On the plus side, the grouping of a facilitator with a collection of client agents provides a natural building block from which to construct larger systems. On the minus side, there is the potential for a facilitator to become a communication bottleneck, or a critical point of failure. In tasks requiring a sequence of exchanges between two agents, it is possible for a facilitator to assist them in finding one another and establishing communication, but then to step out of the way while they communicate over a direct, dedicated channel. This is a relatively straightforward extension to our approach, which we plan to incorporate. For more complex task configurations, we see three general areas to explore in addressing these issues. First, a variety of multifacilitator topologies can be exploited in constructing large systems.

It would be useful to investigate which of these exhibits the most desirable properties with respect to both scalability and fault tolerance. Second, it is possible to modularize the facilitator's key functionalities. For example, goal planning (delegation and optimization) can readily be separated from goal execution. Given this, one can envision a configuration in which the execution task is distributed to other agents, thus freeing up the facilitator. Third, we would like to incorporate mechanisms for basic transaction management, periodically saving the state of agents (both facilitator and client), and rolling back to the latest saved state in the event of the failure of an agent.

With respect to agent development tools, we plan on updating our initial work in this area (described at PAAM96 in (Martin et al., 1996)) to a more group-oriented and web-centric design. Improvements to the linguistic tools, and a graphical monitoring agent would also be desirable.

While much work has been done by agent researchers to demonstrate increased autonomy of individual agents (particularly in the category of information filtering and personal assistants), smarter and more autonomous facilitators (or other means of coordinating multiple agents) are likely to be more critical to the evolution of multiagent systems. Our experience to date has shown value in the handling of compound goals, with advice parameters, by facilitators. However, the advice is still relatively simple, and the discretion exercised by the facilitator relatively limited. Thus, we are interested in exploring the use of more sophisticated strategies by the facilitator, guided by a higher level of advice. It may be possible to draw upon existing work in the (artificial intelligence) field of planning and the (database) field of query planning. Facilitation is also likely to benefit from richer representations of agents' capabilities.

12 Summary

The Open Agent Architecture provides a framework for the construction of distributed software systems, which facilitates the use of cooperative task completion by flexible, dynamic configurations of autonomous agents. We have presented the rationale underlying its design, compared its features to those of other distributed frameworks, and summarized the applications built to date using it. In addition, we have described the major components of OAA infrastructure, and the mechanisms used in assembling an agent-based system. These mechanisms include a general approach to achieving cooperation between agents, organized around the declaration of capabilities by service-providing agents, the construction of goals by users and service-requesting agents, and the role of facilitators in coordinating the satisfaction of these goals, subject to advice and constraints that may accompany them; facilities for creating and maintaining shared repositories of data; and the use of triggers to instantiate commitments within and between agents.

References

Adam Cheyer and Luc Julia. 1995. Multimodal maps: An agent-based approach. In *Proc. of the International Conference on Cooperative Multimodal Communication (CMC/95)*, Eindhoven, The Netherlands, May. Also available at http://www.ai.sri.com/~oaa/ + "Bibliography".

Adam Cheyer and Luc Julia. 1998. Mviews: Multimodal tools for the video analyst. In *Proceedings of the 1998 International Conference on Intelligent User Interfaces (IUI98)*, San Francisco, California, January.

Adam Cheyer, Luc Julia, and Jean-Claude Martin. 1998. A unified framework for constructing multimodal applications. In *Proceedings of the 1998 Conference on Cooperative Multimodal Communication (CMC98)*, San Francisco, California, January.

Philip R. Cohen, Adam J. Cheyer, Michelle Wang, and Soon Cheol Baeg. 1994. An open agent architecture. In O. Etzioni, editor, *Proc. of the AAAI Spring Symposium Series on Software Agents*, pages 1–8, Stanford, California, March. American Association for Artificial Intelligence.

Tim Finin, Yannis Labrou, and James Mayfield. 1997. KQML as an agent communication language. In Jeff Bradshaw, editor, *Software Agents*. MIT Press, Cambridge.

FIPA. 1997. Foundation for intelligent physical agents (FIPA) 1997 specification. Available online at http://drogo.cselt.stet.it/fipa/spec/fipa97.htm.

D. Gelernter. 1993. *Mirror Worlds*. Oxford University Press, New York.

Michael R. Genesereth and Richard E. Fikes. 1992. Knowledge interchange format version 3.0 reference manual. Technical Report Logic-92-1, Stanford University, Stanford, CA. Also available online at http://logic.stanford.edu/kif/kif.html.

M. R. Genesereth and S. P. Katchpel. 1994. Software agents. *Communications of the ACM*, 37(7):48–53.

M. R. Genesereth and N. P. Singh. 1993. A knowledge sharing approach to software interoperation. Technical Report Logic-93-1, Department of Computer Science, Stanford University, Stanford, CA.

Didier Guzzoni, Adam Cheyer, Luc Julia, and Kurt Konolige. 1997. Many robots make short work: Report of the SRI international moble robot team. *AI Magazine*, 18(1):55–64.

Yannis Labrou and Tim Finin. 1997. A proposal for a new KQML specification. Technical Report CS-97-03, Computer Science and Electrical Engineering Department, University of Maryland Baltimore County, Baltimore, MD 21250, February. Also available online at http://www.cs.umbc.edu/kqml/.

David L. Martin, Adam Cheyer, and Gowang-Lo Lee. 1996. Agent development tools for the open agent architecture. In *Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology*, pages 387–404, Blackpool, Lancashire, UK, April. The Practical Application Company Ltd.

David L. Martin, Hiroki Oohama, Douglas Moran, and Adam Cheyer. 1997. Information brokering in an agent architecture. In *Proceedings of the Second International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology*, Blackpool, Lancashire, UK, April. The Practical Application Company Ltd.

Microsoft. 1996. Distributed component object model protocol – DCOM/1.0. Available online at http://www.microsoft.com/activex/ + DCOM.

Robert Moore, John Dowding, Harry Bratt, J.M̃ark Gawron, Yonael Gorfu, and Adam Cheyer. 1996. Commandtalk: A spoken-language interface for battlefield simulation. Technical report, Artificial Intelligence Center, SRI International, 21 June. Also, http://www.ai.sri.com/natural-language/projects/arpa-sls/apps.html.

Douglas B. Moran and Adam J. Cheyer. 1995. Intelligent agent-based user interfaces. In *Proc. of International Workshop on Human Interface Technology 95 (IWHIT'95)*, pages 7–10, Aizu-Wakamatsu, Fukushima, Japan, 12-13 October. The University of Aizu. Also available at http://www.ai.sri.com/∼oaa/ + "Bibliography . . . ".

Douglas B. Moran, Adam J. Cheyer, Luc E. Julia, and David L. Martin. 1997. The open agent architecture and its multimodal user interface. In *Proceedings of the 1997 International Conference on Intelligent User Interfaces (IUI97)*, Orlando, Florida, 6-9 January.

M. H. Nodine and A. Unruh. 1997. Facilitating open communication in agent systems: the InfoSleuth infrastructure. Technical Report MCC-INSL-056-97, Microelectronics and Computer Technology Corporation, Austin, Texas 78759, April.

Object Management Group (OMG). 1997. The complete CORBA/IIOP 2.1 specification. Available online at http://www.omg.org/corba/corbiiop.htm.

A. Rao and M. Georgeff. 1995. BDI agents from theory to practice. Technical Note 56, AAII, April.

David G. Schwartz. 1995. *Cooperating Heterogeneous Systems.* Kluwer Academic Publishers, Dordrecht.

K. Sycara, K. Decker, and M. Williamson. 1996. Matchmaking and brokering. In *Proc. of the Second International Conference on Multi-Agent Systems (ICMAS-96)*, December.

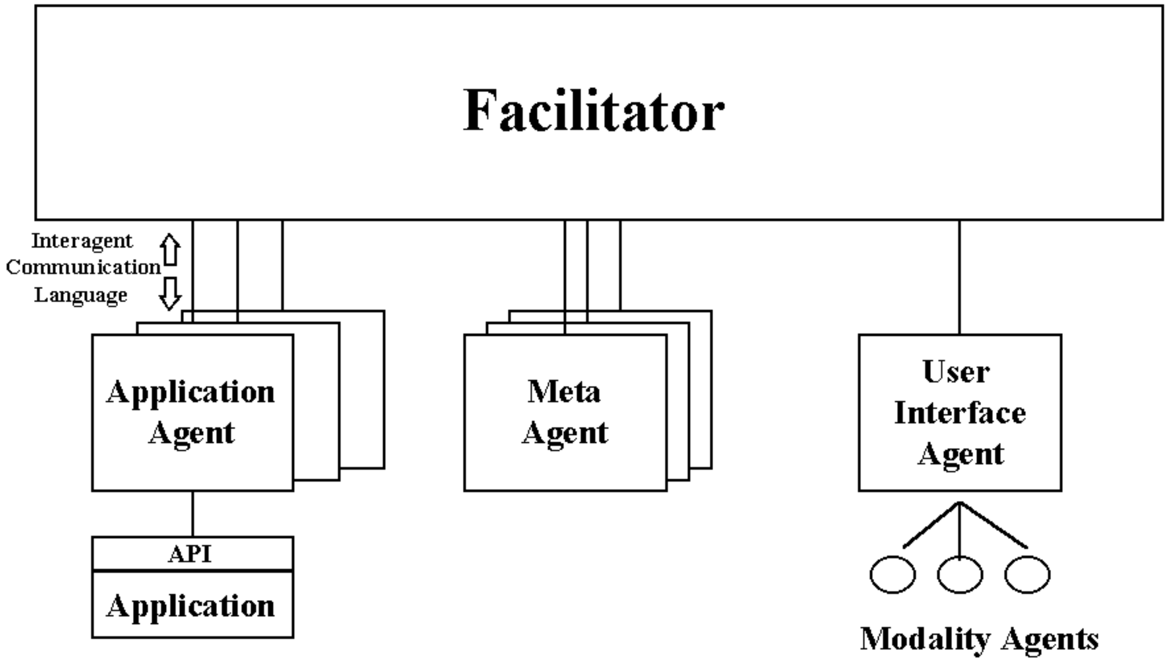


Figure 1: OAA System Structure.

Application	**Description**
Automated Office	Mobile interfaces (PDA with telephone) to integrated community of commercial office applications (calendar, database, email) and AI technologies (speech recognition, speaker identification, text to speech, natural language interpretation and generation). (Cohen et al., 1994)
Unified Messaging	Adaptable, ubiquitous access to email, fax, voice, and Web messages and services
Multimodal Map	Pen/voice interface to distributed Web data. (Cheyer and Julia, 1995)
InfoWiz	Animated voice interactive interface to the Web.
ATIS-Web	Try out a live demo of speech recognition over the Web! Available at http://www.speech.sri.com/demos/atis.html
CommandTalk	Spoken-language interface for controlling simulated forces. (Moore et al., 1996)
Spoken Dialog Summarization	Real-time system for summarizing human-human spontaneous spoken dialogs (Japanese).
Language Tutoring	Speech recognition for foreign language learning, incorporating user modeling for adaptive lessons.
Disaster response	Collaborative, wireless map-based interface for emergency response teams.
MVIEWS	Integrating speech, pen, natural language, image processing and other technologies for the video analyst. (Cheyer and Julia, 1998)
OAA InfoBroker	Mediated facilitation of heterogeneous structured and semistructured (Web) datasources. (Martin et al., 1997)
OAA Rental Agent	Monitors the Web and notifies user when housing classifieds meet user specifications.
Agent Development Tools	Guides the agent developer through the steps required to create new agents. (Martin et al., 1996)
Multi-Robot Control	Team of robots works together on assigned tasks (1st place, AAAI Office Navigation Event). (Guzzoni et al., 1997)
Surgical Telepresence	Force feedback training simulator for endoscopic surgery. All physical and virtual entities modeled as OAA agents.

Table 1: A partial list of applications written using OAA.

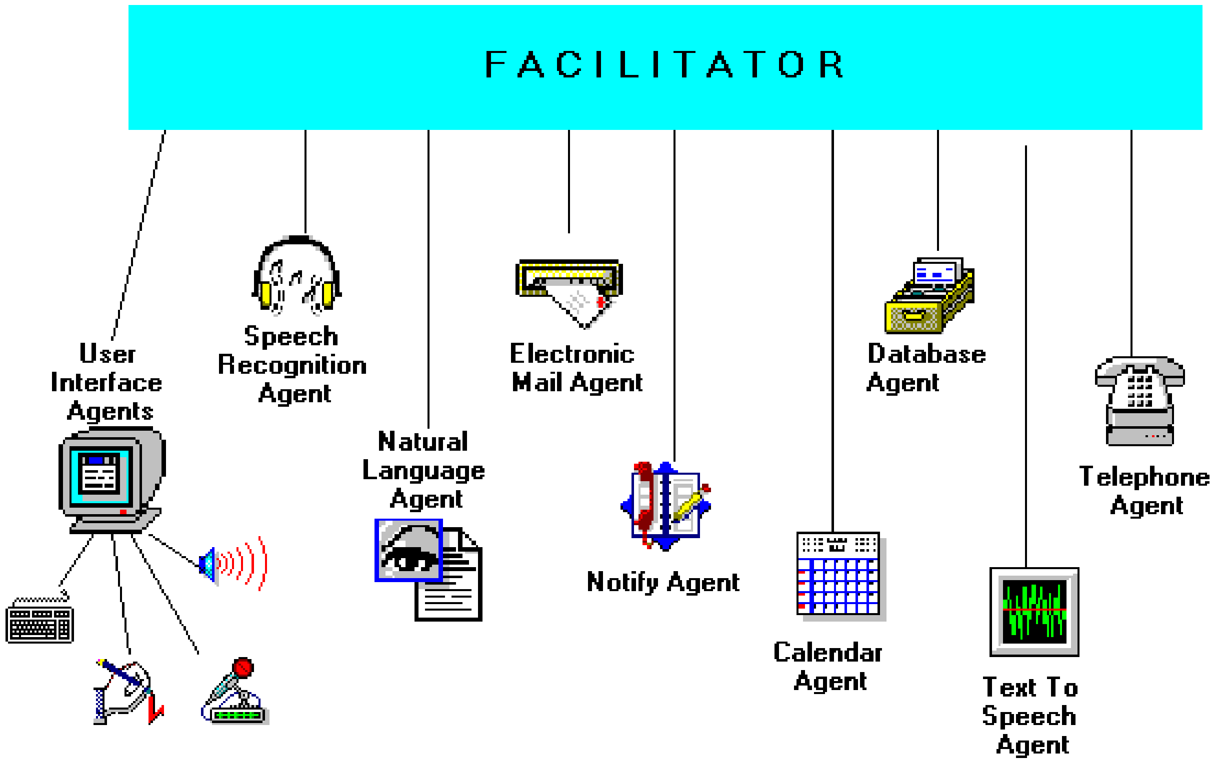


Figure 2: Automated Office Agents.

Figure 3: User Interface for Automated Office Application.

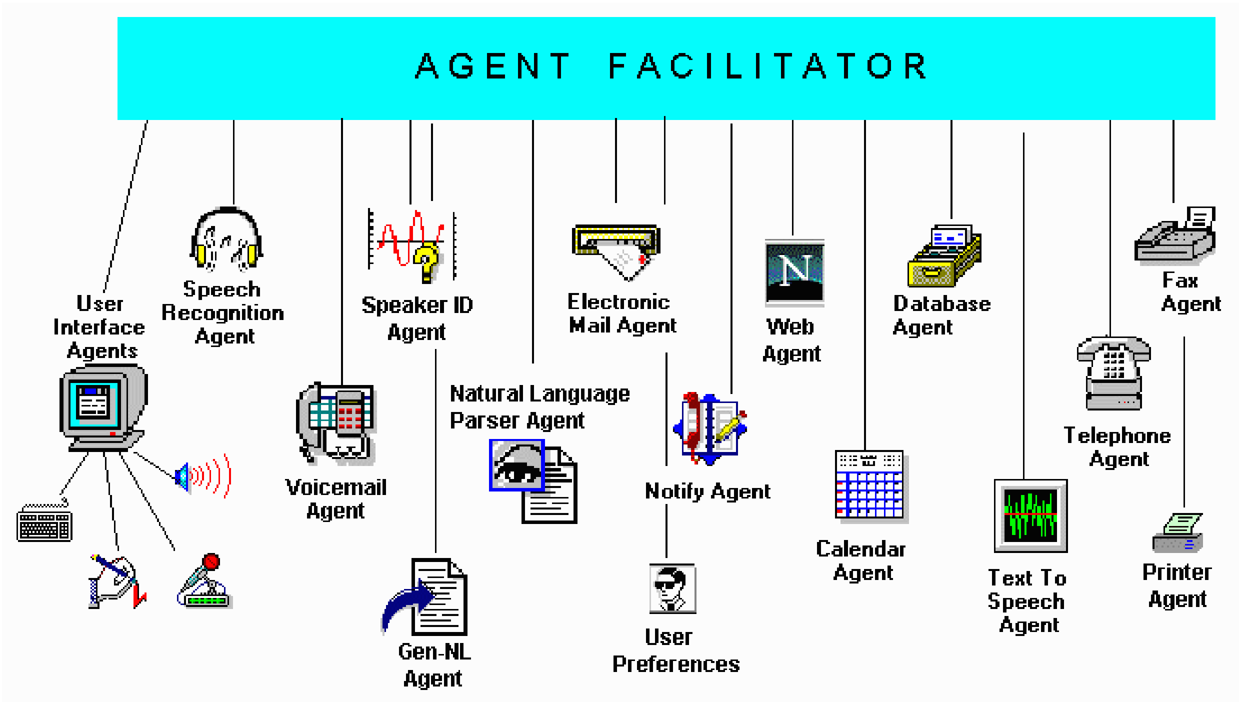


Figure 4: Unified Messaging Agents.

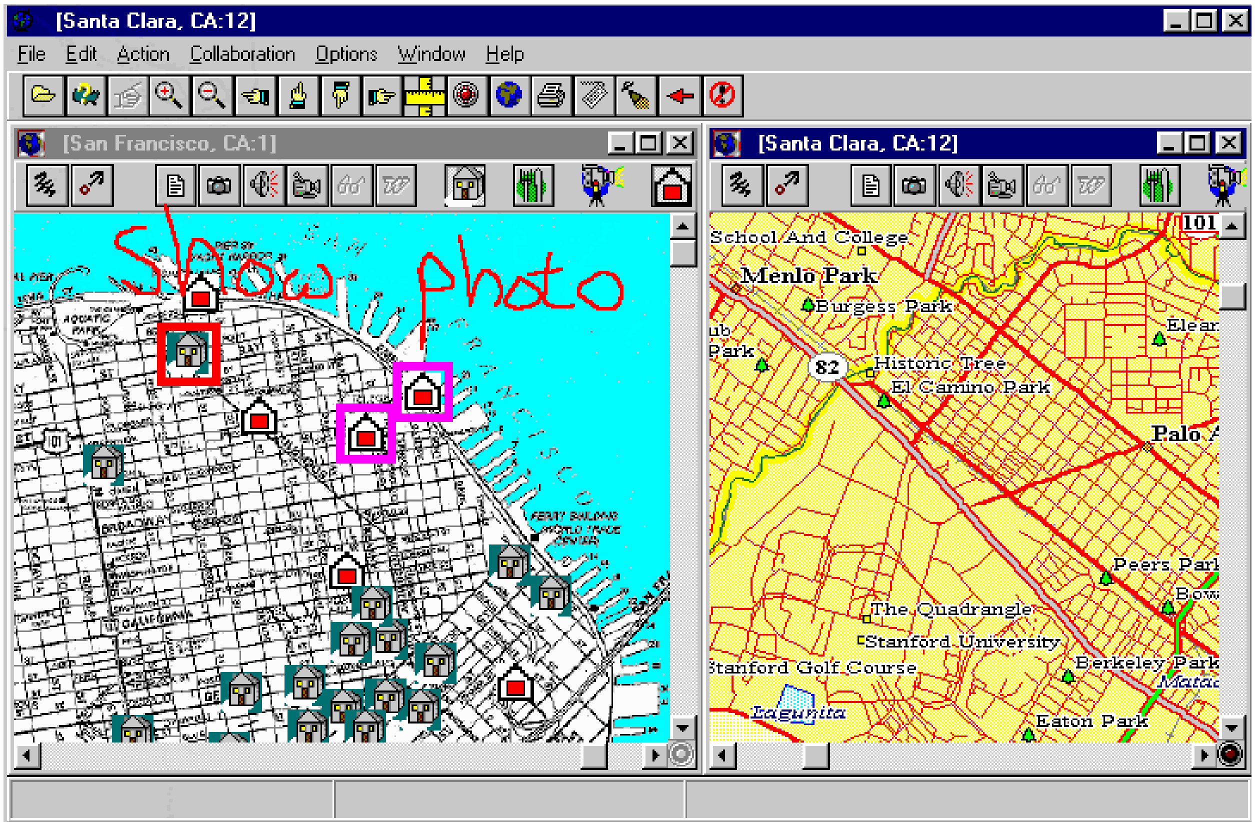


Figure 5: Multimodal Map Application.

Exhibit E

OAA�-Based Applications

More than 35 applications have been implemented using the Open Agent Architecture. Here is a partial list to give you ideas!

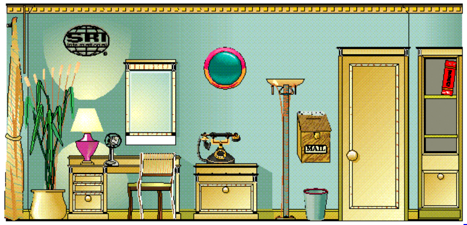
OAA-based Applications

- Automated Office: Agents automate access to DBs,calendar, etc.
- Multimodal Map: Pen/Voice interface to distributed web data.
- CommandTalk: Spoken-language interface for controlling simulated forces.
- ATIS: Try out a **live demo** of speech recognition over the web!
- Agent Development Tools: Simplifies the creation of new OAA agents.
- Multi-Robot Control: A team of robots work together on assigned tasks.
- Spoken Dialog Summarization
- InfoWiz: An animated voice interactive interface to the web.
- MVIEWS: Integrating speech, pen, NL, image processing and other technologies for the video analyst.
- MAESTRO: A system for coordinating multiple recognition technologies (speech, OCR, NL Extraction, etc.) for indexing broadcast news videos.
- SUO Robots: Semi-autonomous robot control using speech and gestures.

  The following OAA-based applications were created by SRI's Computer Human Interaction Center (CHIC!). CHIC!, which was active during 1998 -- 2000, explored new ways that people will interact with mediated spaces, smart environments, and e-services.

- CHeF: The Collaborative Home e-Fridge (CHeF).
- OfficeMate: A wireless, position-sensitive slate-computer for augmenting a visitor's experience to SRI.
- TravelMate and CARS: Augmenting the driving experience through speech recognition and a "smart windshield".
- EMCE: The Enhance Multimodal Collaborative Environment is a great place for meetings.
- SURF: An interface for controlling your OAA-enabled household appliances from your TV!

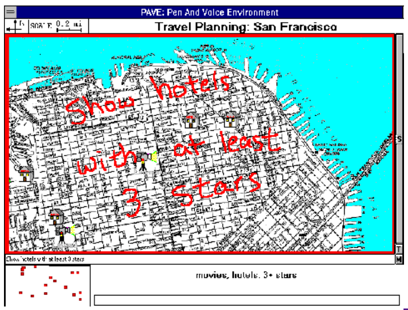
Perhaps the most exciting aspect of OAA applications is the cross-application possibilities: for instance, the agent community making up CHeF the smart fridge, can communicate with agents in your car, your house, your office, and so forth. Where others build standalone applications, these CHIC! applications use OAA to explore the truly *connected world!*



Automated Office

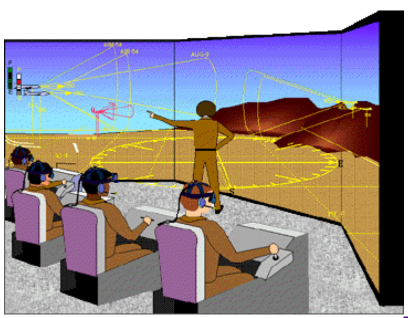
Control your office from remote locations -- agents provide access and monitoring of your calendar, email, or database applications over the telephone, a laptop, web browser or wireless PDA.

- Paper: (AAAI 1994). HTML version, Compressed PostScript file (.gz), 55K
- Office Video (GUI+Phone) (AVI -- 7.5Mb), (AVI.zip -- 2.8Mb)
- Phone-Only GUI (AVI -- 6.4Mb), (AVI.zip -- 1.8Mb)



Multimodal Maps

A teacher at a blackboard communicates with a classroom of students by drawing, speaking, underlining, circling, writing -- all simultaneously. Communicating with a network of distributed software agents should be this easy! Using a map-style interface, users can naturally combine writing, speaking, and gesturing to retrieve information and issue commands to distributed agents.



CommandTalk

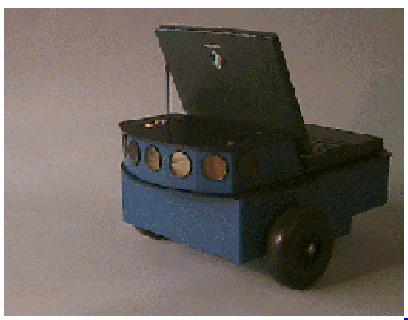
Related to the multimodal map project, CommandTalk is a spoken-language interface that allows commanders to use natural, spoken English commands to control simulated forces. This system is in use at the Marine Corps Air Ground Combat Center, Twentynine Palms, California.

[Speech Recognition over the Web]

Try it!!!
Use your telephone and Java-enabled web browser to talk to an OAA-based demonstration application for scheduling airline reservations.

Agent Development Tools

A set of Agent Development Tools, implemented within the OAA themselves, guide a programmer through the tasks involved in implementing new OAA agents and applications.



Multi-Robot Control

OAA-powered robots wins **first place** at 1996 AAAI Robotics Competition and Exibition (Office Navigation Task).

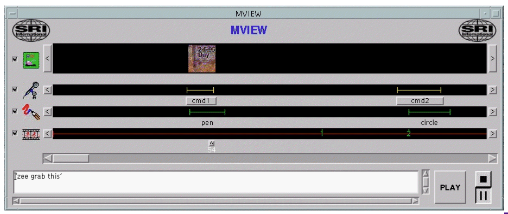
- Paper: GUZZONI D., CHEYER A., JULIA L. & KONOLIGE K. (1997). *Many Robots Make Short Work. Report of the SRI International mobile robot team at AAAI96* AI Magazine, Spring 97, pp 55-64. (PS: 186 Kb)
- Video (Real Video -- 2.6Mb)

Spoken Dialog Summarization

A real-time system for summarizing human-human spontaneous spoken dialogues (Japanese).

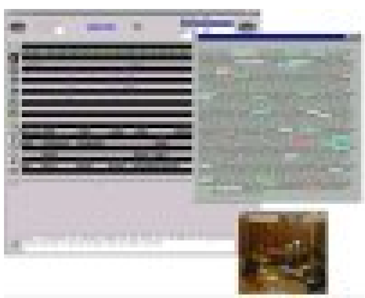
 [InfoWiz]

An animated voice interactive system for learning about SRI.



[MVIEWS]

MVIEWS enables an analyst to annotate a live video stream using pen and voice, and incorporates various technologies (pen & speech recognition, image processing, content-based indexing, collaboration) to enhance the analyst's abilities to intelligently monitor the situation.

 [MAESTRO]

A system for coordinating multiple recognition technologies (speech, OCR, NL Extraction, etc.) for indexing broadcast news videos.

- Paper (PDF: CACM 1999)
- Video (Real Video -- 1Mb)

SUO Robots

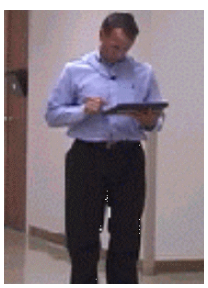
This project focused on a wireless multimodal interface for controlling teams of robots and their sensors (including video) in Small Unit Operations (SUO) missions.

- Video (AVI -- 135Mb!)

CHEF

Among other things, this smart fridge knows what's inside and can find recommended recipes using the ingredients.

- Brochure (PDF)
- Video (Real Video -- 3.66Mb)



OfficeMATE

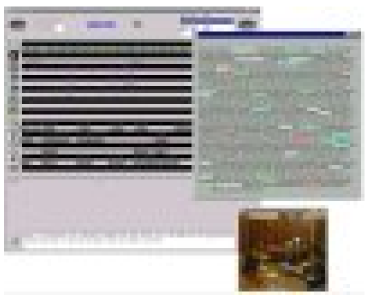
The objective is to give SRI visitors a wireless tablet computer that augments their experience during the day. Position tracking helps the user navigate SRI hallways and resources, and information is updated on the display based on the user's location and interests.

- Video (Real Video -- 2.55Mb)

TravelMATE and CARS

TravelMATE uses GPS, a compass, a 3D virtual reality terrain database and a smart windshield to augment a tourist's driving experience with context-relevant information. CARS provides a speech-enabled interface to systems and services provided by the car and by the Internet.

- Brochure (PDF)
- Video (Real Video -- 2.83Mb)



EMCE

EMCE, the Enhanced Multimodal Collaborative Environment, plays host during augmented meetings.

- Brochure (PDF)

SURF

Surf provides an OAA-based interface for your TV, where you can task your community of agents (home appliances and information services) by speaking into your remote control. Events (e.g. the phone ringing) can also trigger informational updates on the TV screen (e.g. the phone number using caller-id).

- Brochure (PDF)

Exhibit F

PAAM '98 Tutorial

Building and Using Practical Agent Applications

David Martin
Adam Cheyer

SRI International

Contents

- Context: Agents & Distributed Computing
- Challenges & Opportunities
- Inside the Open Agent Architecture
- Example Systems & Useful Techniques
- Concluding Remarks

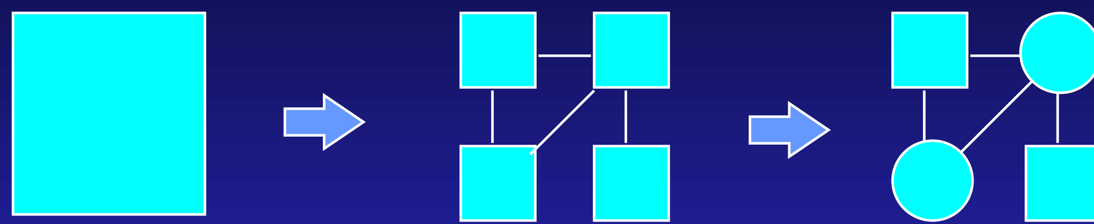
- **Context: Agents & Distributed Computing**
  - **Areas of Agent Research**
  - **Evolving Paradigms for Distributed Systems**
  - **SRI's Open Agent Architecture**
- **Challenges & Opportunities**
- **Inside the Open Agent Architecture**
- **Example Systems & Useful Techniques**
- **Concluding Remarks**

What Is an Agent?

Examples

Examples	
Voyager, Aglets, Odyssey	Mobile Agents Programs that move among computer hosts
Robots, Softbots, BDI	Autonomous Agents Based on planning technologies
FireFly, MIT Media Lab	Learning Agents User preferences, collaborative filtering,...
Microsoft Agent, Julia	Animated Interface Agents Avatars, chatbots, ...
ModSAF, RoboCup	Simulation-based Entities
SIMS, InfoSleuth, IR	Data/Info finding, filtering and merging
OAA, KQML, FIPA	Cooperative Agents Cooperation among distributed heterogeneous programmatic components

Approaches to Building Applications



Monolithic Applications

Object-Oriented Applications

Distributed Object Applications

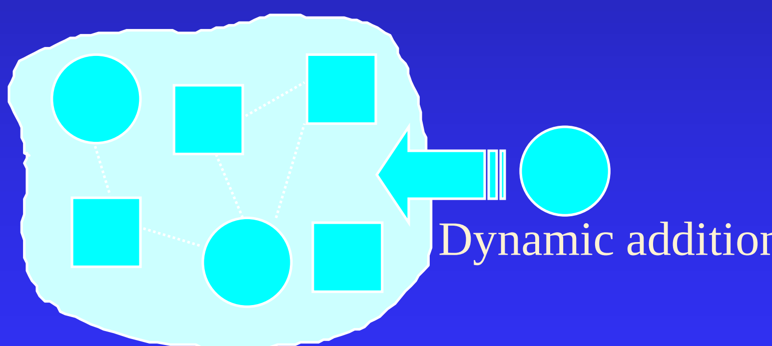
Objective

Suitable for Internet environment

Virtual community of dynamic services

Adaptable to changing, evolving network resources

Flexible interactions among components



OAA Applications

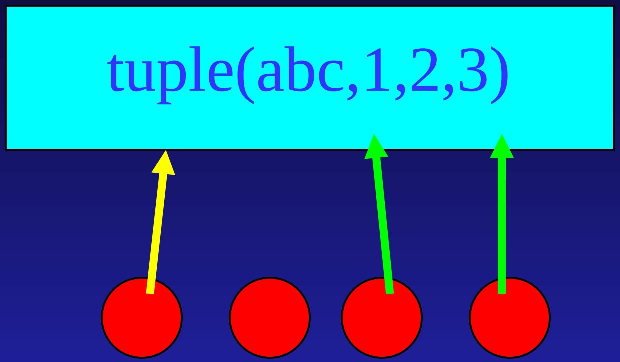
Approaches to Distributed Computing

- Mobile Objects
- Blackboard Architectures
- Agent Communication Languages (ACL)
- Publish & Subscribe Brokers

Mobile Objects (Agents)

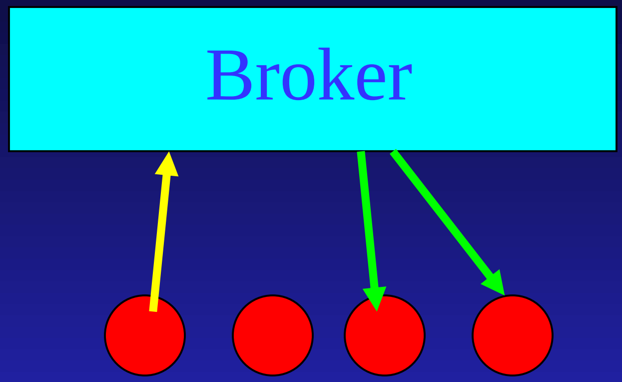
- Objects move under their own power (e.g., Voyager, Aglets)
- Advantages
  - Network bandwidth for certain classes of problems
  - Parallelism - many objects can be spawned
- Disadvantages
  - Programmatically specify where to go and what to do, through a known interface
  - Little automated support for inter-object cooperation
  - Programming language specific (non-heterogeneous)

Blackboard Architectures



- **Knowledge Sources read and write tuples from a common information space (e.g. LINDA, FLiPSiDE)**
- **Advantages**
  - **Eliminates explicitly programmed interactions among participants**
- **Disadvantages**
  - **KS cannot coordinate interactions**
  - **Polling**

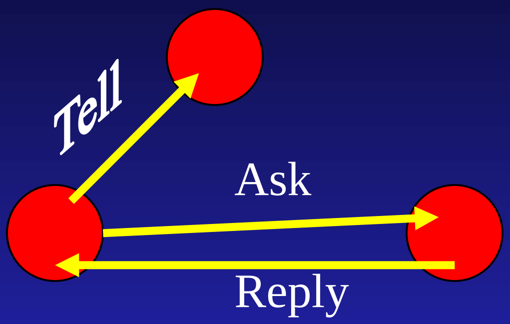
Publish & Subscribe Brokers



- Clients register interest, broker routes/filters msgs
- Examples: Talarian SmartSockets, Active Software's ActiveWeb, ACL Brokers
- Advantages
  - Destination process(es) not explicitly encoded
  - No polling
- Disadvantages
  - Simple filtering, unitary messages

Agent Communication Languages

ANS, Service Broker



- **Communication message types based on speech acts (e.g., ask, tell, deny) + conversational policies**
- **Examples: FIPA ACL, KQML**
- **Advantages**
  - **Rich interaction model, peer-to-peer based**
  - **Standardized message types, content-agnostic**
- **Disadvantages**
  - **Conformance to specs not universal**
  - **Explicitly coded interactions among participants**

Comparison of Distributed Approaches

Distributed	Dist. Objects, Mobile Agents, ACL, Blackboards, Pub/Sub
Heterogeneous languages	Distributed Objects, ACL, Blackboards
Non-coded interactions	Blackboards, Pub/Sub
Parallel Services	Blackboards
Compound Expressions	(Mobile Agents)
Constraints	No

Overview of the OAA

Definition

OAA: A framework for integrating a community of software agents in a distributed environment

Distributed Computing Through Delegation:

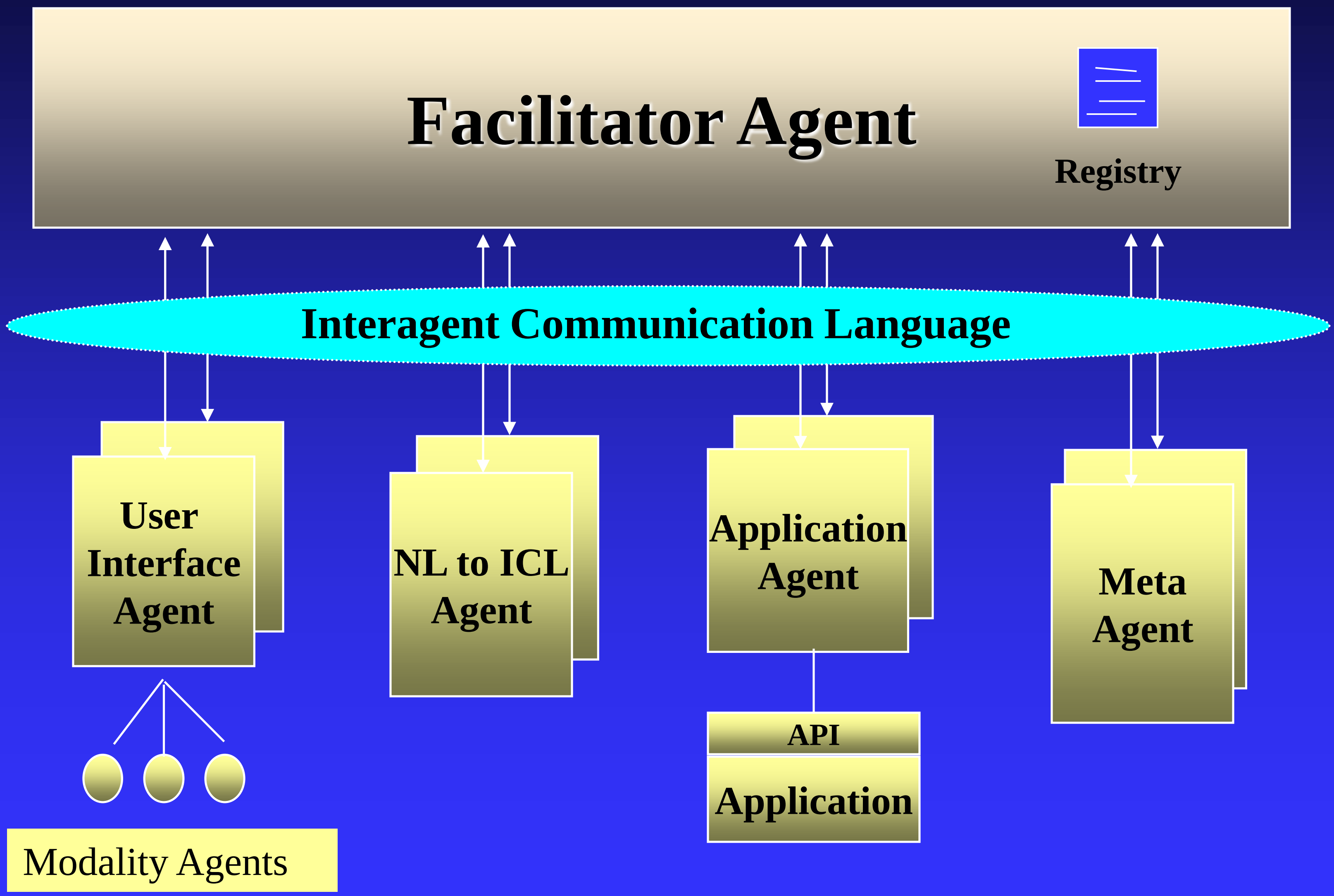
What, not how or who

- Facilitates flexible, adaptable interactions among distributed components through *delegation* of tasks, data requests & triggers

User Interface

- Enables natural, mobile, multimodal user interfaces to distributed services

OAA Architecture



Main Points Automated Office Application

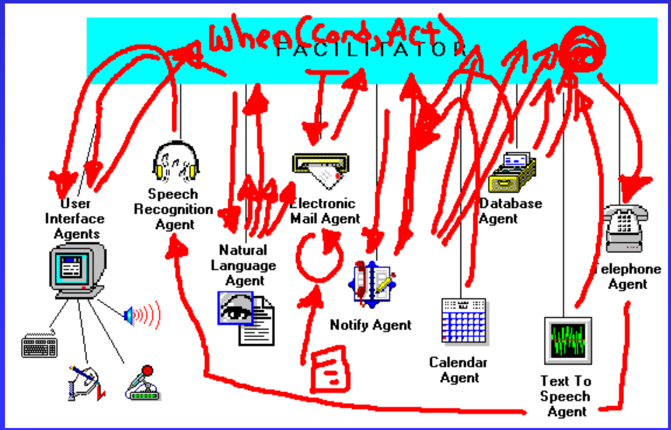
Mobile access to distributed services

Legacy applications interacting with AI technologies

High-level tasking of agents through NL and speech

Flexible interactions among components

Delegated Triggers



OAA Characteristics

Open: agents can be created in many languages and interface with existing systems

Extensible: agents can be added or replaced dynamically

Distributed: agents are spread across many computers

Parallel: Parallel execution of subtasks

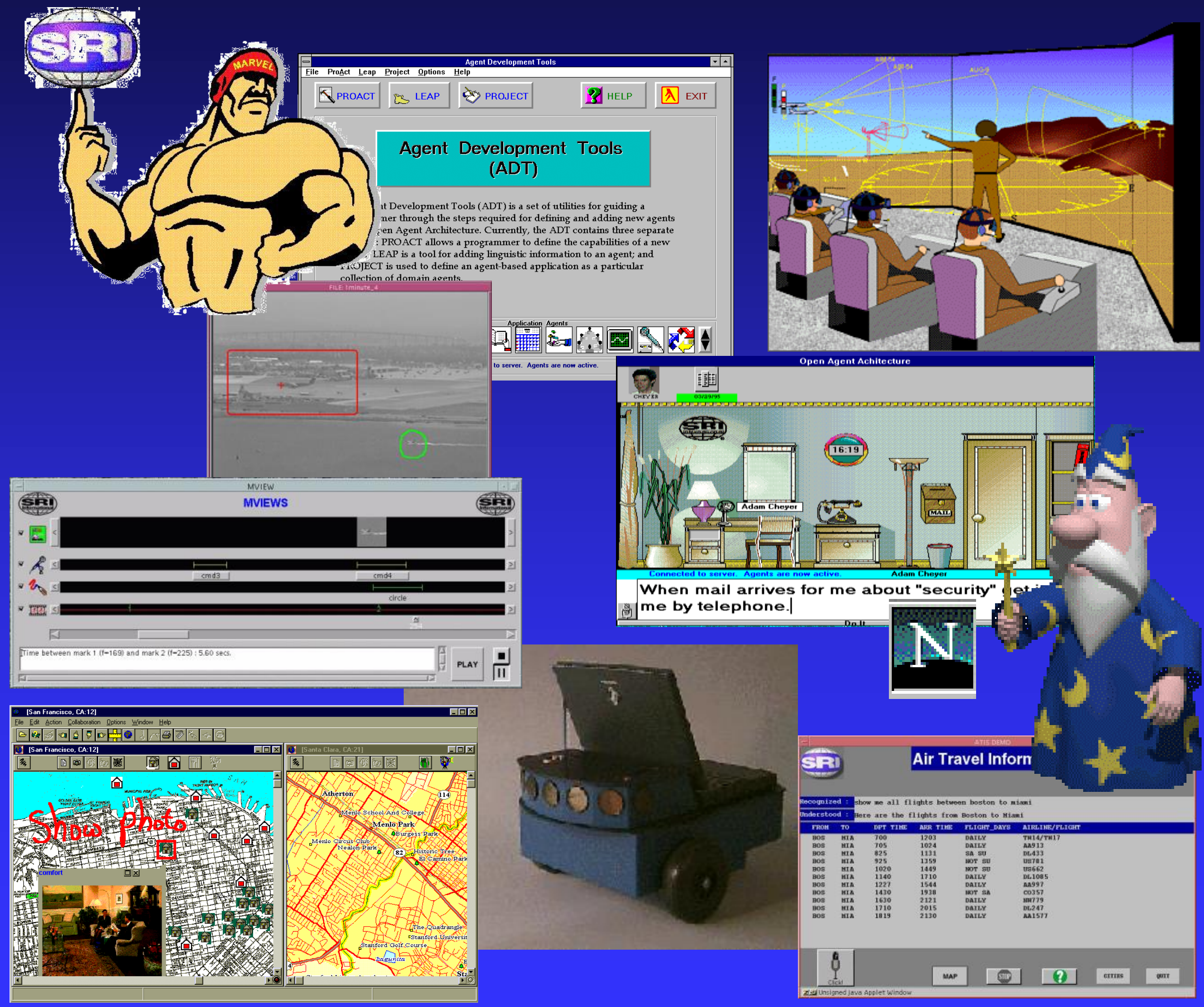
Mobile: Lightweight interfaces on phone and/or PDA

High-level: hides software and hardware dependencies

Multimodal: handwriting, speech, gestures, and direct manipulation can be combined together

OAA-based Applications

1. Automated Office
2. Unified Messaging
3. Multimodal Maps
4. CommandTalk
5. ATIS-Web
6. Spoken Dialog Summarization
7. Agent Development Tools
8. InfoBroker
9. Rental Finder
10. InfoWiz Kiosk
11. Multi-Robot Control
12. MVIEWS Video Tools
13. MARVEL
14. SOLVIT
15. Surgical Training
16. Instant Collaboration
17. Crisis Response
18. WebGrader
19. Speech Translation
20-25+ ...



- Context: Agents & Distributed Computing
- Challenges & Opportunities
  - Interoperation
  - Coordination & Control
  - Information Management and Sharing
  - Intelligent User Interfaces
- Inside the Open Agent Architecture
- Example Systems & Useful Techniques
- Concluding Remarks

Interoperation

- Language, Ontology, Conversational Protocol
- Imposing the Right Amount of Structure
- Legacy & “Owned-elsewhere” Applications
  - Multi-platform, Multi-language
  - Wrappers & Surrogates
  - Backwards Compatibility With Older Paradigms
- Integration with Standards
- Opportunities
  - Support Greater Flexibility & Dynamism in Structuring Communities & Interactions
  - Provide Economical Means of Coding Interactions
  - Leverage Our Understanding Of Conversations
  - Minimize Platform & Language Barriers

Coordination & Control

- No Fully General Solutions Available
- Families of C & C Strategies
  - Knowledge-Sharing
  - Team Coordination
  - Economic (Market-Driven)
  - Evolutionary
- Opportunities
  - Flexibility, Synergy
  - Advice and Constraints
  - Temporal Control
  - Sophisticated Facilitation, Reactive Execution

Alternative Agent Control Strategies

- **Knowledge-Sharing**
  - Agents share knowledge about capabilities and requests.
  - Agent *brokers* dynamically match requests to capabilities.
  - System dynamically adjusts as capabilities are added to and removed from the environment.

- **Team Coordination**
  - Agents share knowledge about goals, plans, tasks & subtasks, commitments and performance.
  - Teams cooperative through partially synchronized actions to accomplish individual subtasks and common goals.

- **Market-Driven Economy**
  - Self-interested agents pursue personal profit.
  - Behavior is driven by the cost of resources.
  - Agents are controlled by specifying market rules, rewards and penalties.

- **Evolutionary Systems**
  - Agents populations evolve over time through “reproduction”, mutation and natural selection.
  - Agents are controlled by specifying selection criteria and reproduction process.

Applicability of Strategies

DARPA

Coordination and Control Strategies

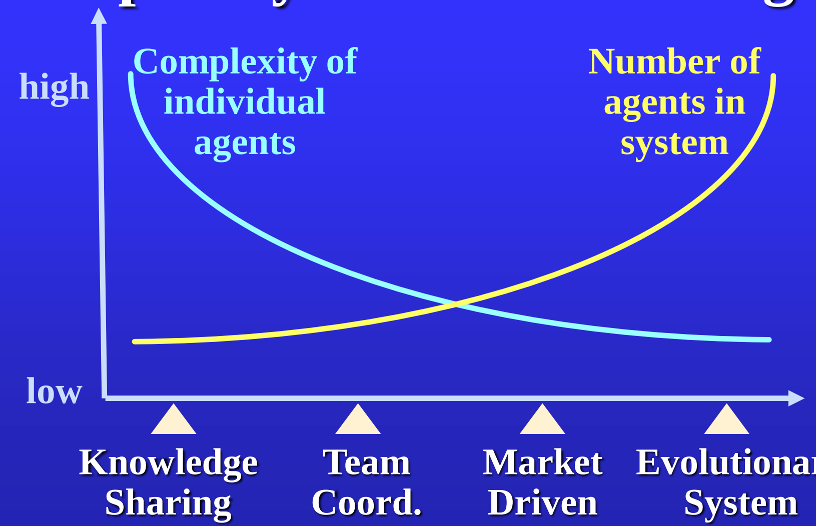
- Knowledge-Sharing
- Team Coordination
- Market-Driven Economy
- Evolutionary Systems

$10^1 - 10^3$

$10^5 - 10^7$

The strategies differ in the complexity and number of agents for which they are suited to control

Complexity vs. Number of Agents



Information Management and Sharing

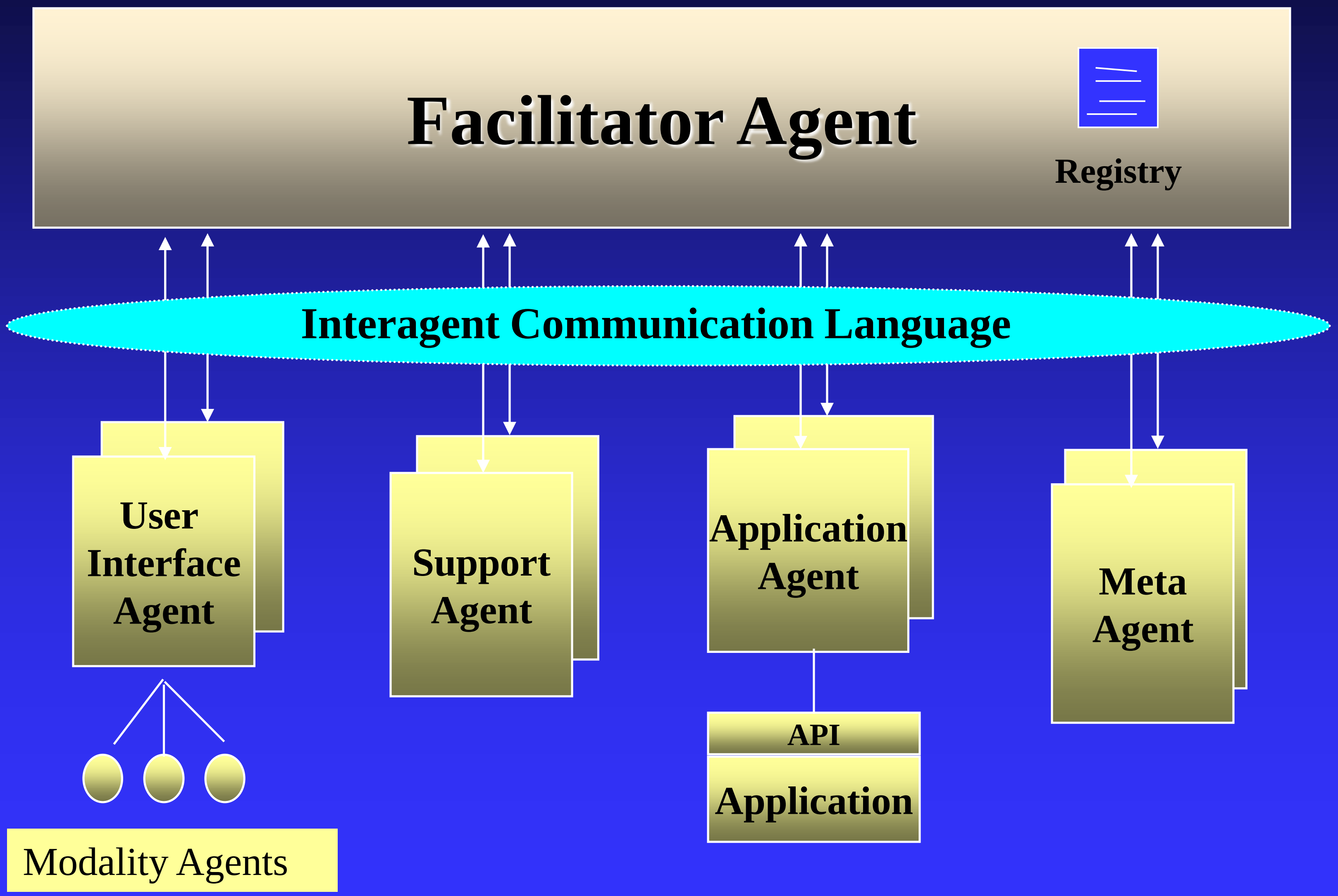
- External Data
  - Heterogeneous, Dynamic, Unreliable Sources
- Operational Data
  - Maintaining Consistent World-views
  - Transactions, Snapshots, Roll-back
- Sharing Strategies
  - How Much to Share, Cost of Sharing
  - Support for Collaboration
- Opportunities
  - Tight Integration With Service-providing & Requesting Mechanisms
  - Built-in Support for Handling Dynamism
  - Use Intelligence, Autonomy to Address Reliability

Intelligent User Interfaces

- Make User Requests Comprehensible to System
- Make System Results Comprehensible to User
- Help User Understand System Complexity …
  - Multiple Autonomous Actors
  - Dynamic Communities
- … Or Not Be Required to
- Opportunities
  - Agent-based Approaches to UI Implementation
  - Integrate Multimodality
  - User As Privileged Member of Agent Community
  - Use of Mixed-initiative Interactions
  - Collaboration

- Context: Agents & Distributed Computing
- Opportunities & Challenges
- Inside the Open Agent Architecture
- Example Systems & Useful Techniques
- Concluding Remarks

OAA Architecture



Interagent Communication Language

- Used by Agents to:
  - Declare Capabilities
  - Request Services of Community
  - Respond to Requests from Other Agents
  - Manage and Exchange Information
- Conversation & Content Layers
- Advice/Constraints Can Accompany Requests
- Platform- and Language-Independence

Providing Services

- Declaring Capabilities
  - solvable(Goal, Parameters, Permissions)
- Examples of Parameters
  - *type*: {data, procedure}
  - *private*: Boolean
  - *utility*: [0 .. 10]

```
solvable(send_message(email, +ToPerson, +Params),
               [type(procedure), callback(send_mail)],
               []),
solvable(last_message(email, -MessageId),
               [type(data), single_value(true)],
               [write(true)])
```

Requesting Services

Task Management

oaa_Solve(*TaskExpr*, *ParamList*)

Expressions: logic-based (cf. Prolog)

Parameters: provide advice & constraints

- *High-level task types*: query, action, inform, ...
- *Low-level*: solution_limit(N), time_limit(T), parallel_ok(TF), priority(P), address(Agt), reply(Mode), block(TF), collect(Mode), ...

Data & Trigger Management

oaa_AddData(*DataExpr*, *ParamList*)

oaa_AddTrigger(*Typ*,*Cond*,*Action*,*Ps*)

Example

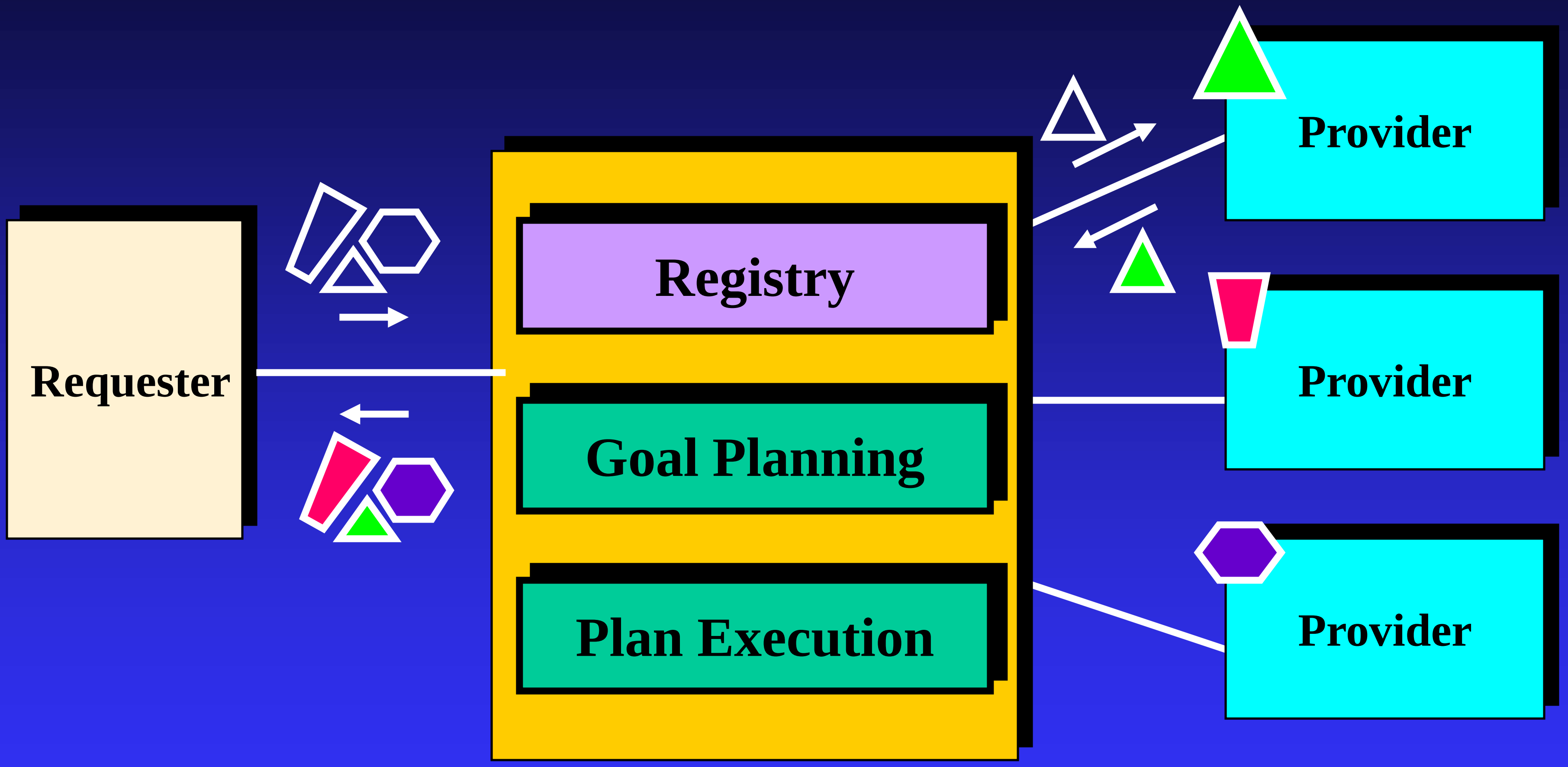
```
oaa_Solve((manager('John Bear',M),
           phone_number(M,P)),
      [query(var(P))])
```

Compound Queries

- Address:Goal::Parameters
  - Address & Parameters Optional
  - Value-returning Parameters
- Composable Using Standard Prolog Operators
- Extensions
  - Parallel Disjunction

```
oaa_Solve(
     (locate('Adam Cheyer', Where)::[strategy(query)],
      notify(MsgRef, 'Adam Cheyer',
               [at(Where), by(fax)])::[strategy(action)]),
     [])
```

Facilitation



Facilitator

OAA Data Management

- Declaring & Utilizing Data Solvables
- Built-in Support
- Example Parameters
  - single_value(t_f), unique_values(t_f)
  - bookkeeping(t_f), persistent(t_f)
  - synonym(Synonym, Original)
  - rules_ok(t_f)
- Maintaining Data Solvables
- Sharing Data

OAA Triggers

Purpose OAA agents can dynamically register interest in any data change, communication event, or real-world occurrence accessible by any agent.

Adding a Trigger oaa_AddTrigger(Type, Cond, Action, Params)

Trigger Types

comm: *on_send*, *on_receive* message
time: "in ten minutes", "every day at 5pm"
data: *on_change*, *on_remove*, *on_add*
task: "when mail arrives about..."

Actions The actions of triggers may be any ICL expression solvable by the community of agents

System-Building Infrastructure

- The Event Loop
- Event Types
  - Built-In
  - Task-Specific
  - Hybrid
- Libraries
  - Multiple Languages Supported
  - Minimal Structure Imposed on Agents

A Sample Text-to-Speech Agent in C

Include libraries

```
#include <libcom_tcp.h>
#include <liboaa.h>
```

List capabilities

```
ICLTerm capabilities = icl_TermFromStr("[play(tts, Msg)]");
```

Define capabilities

```
ICLTerm oaa_AppDoEvent(ICLTerm Event, ICLTerm Params) {
    if (strcmp(icl_Str(Event), "play") == 0) {
        return playTTS(icl_ArgumentAsStr(Event, 2));
    }
    else return NULL;
}
```

Agent Startup

```
main() {
    com_Connect("parent", connectionInfo);
    oaa_Register("parent", "tts", capabilities);
    oaa_MainLoop(True);
}
```

A Sample Text-to-Speech Agent in Prolog

Include libraries

```
:- use_module(com).
:- use_module(oaa).
```

List capabilities

```
capabilities([
    solvable(play(tts, Msg),
            [type(procedure), callback(tts_events)],
            [])]).
```

Define capabilities

```
tts_events(play(tts, Msg), Params) :-
        tts_api(Msg).
```

Agent Startup

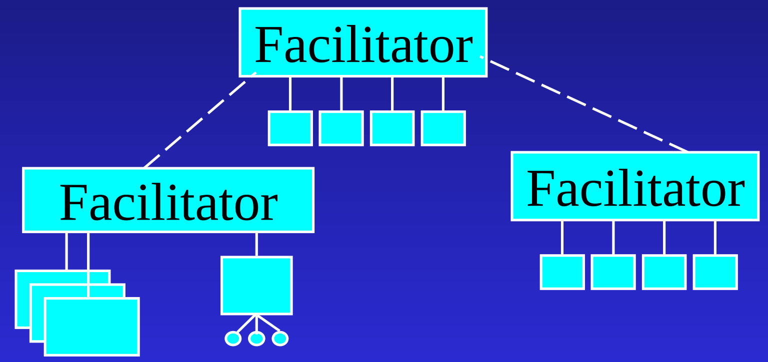
```
start :-
        capabilities(C),
        com_Connect(parent, ConnectionInfo),
        oaa_Register(parent, tts, C),
        oaa_MainLoop(true).
```

OAA and Scalability

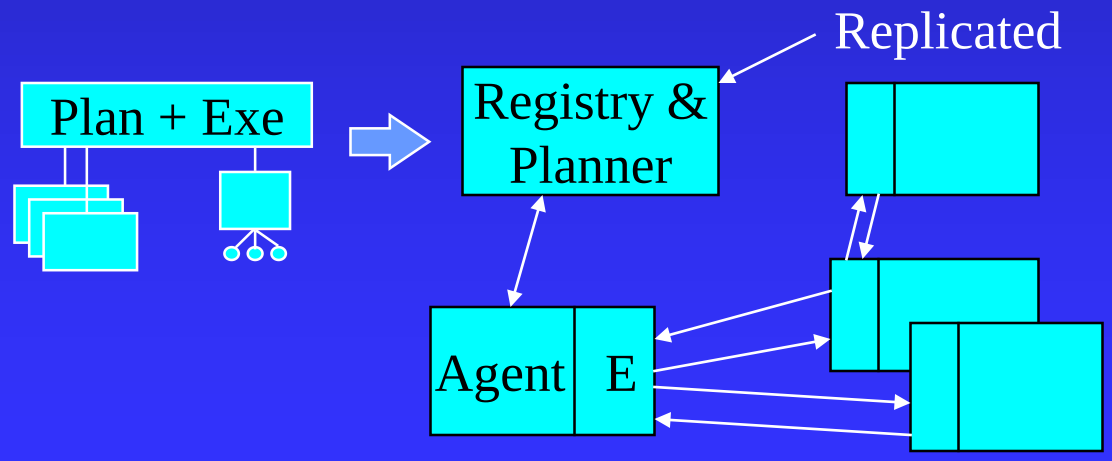
Limitations: Facilitator is single point of failure
Facilitator is bottleneck for communication

Solutions?

Multi-Facilitator topologies



Distribution of planning & execution functions of Facilitator + peer-to-peer communication



- Context: Agent Types & Approaches
- Challenges & Opportunities
- Inside the Open Agent Architecture
- Example Systems & Useful Techniques
  - Agent & Interagent Programming Tips
  - Dynamic Presentation: Unified Messaging
  - Reference Resolution: Multimodal Map
  - Information Management and Collaboration: InfoBroker & Multimodal Map
  - Incremental System Development & Evaluation: Stimulate
  - Looking for the Killer App: Other Tries
- Concluding Remarks

Agent & Interagent Programming Tips

- Choosing an Agent Interface
- Information Sharing Strategies
- Domain-Specific vs. Domain-Independent Agents
- Adding Speech & NL to Interfaces

Tips: Choosing an Agent Interface

- Natural-language inspired interfaces
  - Imperative Verb, Direct Object, ParamList, (Result)
  - Parameter lists hold Adjs, Advs & Prepositions as well as extensible programmatic instruction

- Classes tagged by type
  - inform(phone, ringing, Params)
  - send_message(MsgRef, Params) :-
    memberchk(by(fax), Params)

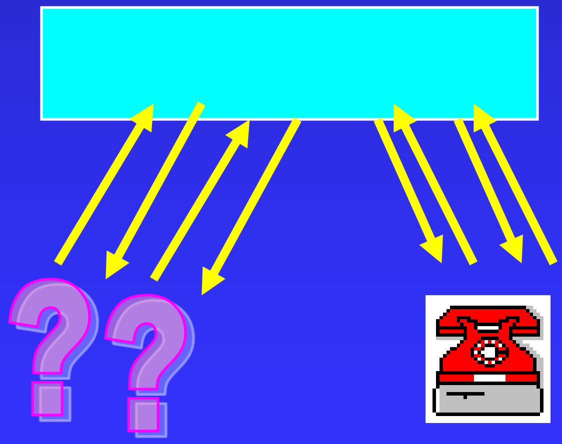
- Succeed once with list vs. Multiple success
  - get(email, message_headers, +Params, -ListOfHeaders)
  - phone_number(Person, PhoneNum)

Tips: 3 Information Sharing Strategies

- Example: Phone Dialer Agent
- <u>1. Query</u>
  - When an agent wants to know the status of the phone, it asks the Facilitator who asks the phone agent

pa: oaa_Declare(status(phone, S),[])

?a: oaa_Solve(status(phone, S), [])

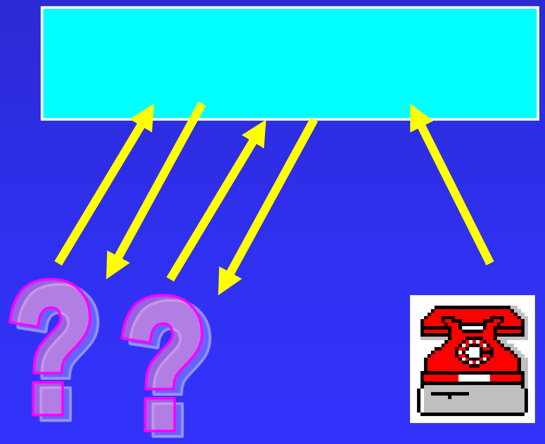


Tips: Information Sharing Strategies - Post

- **2. Post (Blackboard)**
  - **The phone agent writes its status to the Facilitator; agents can query the facilitator for status, and install a trigger which proactively monitors changes to status**

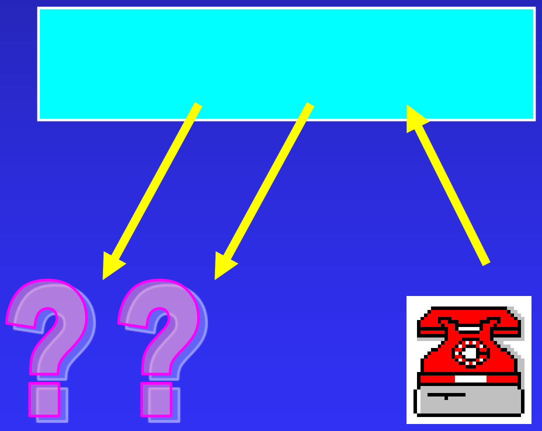
pa: oaa_AddData(status(phone, busy), [])

ia: oaa_Solve(status(phone, S), []),
oaa_AddTrigger(data, status(phone,S), notify(Me, phone(S)), [on(change)])



Tips: Information Sharing Strategies - Inform

- **3. Inform**
  - **Broadcast time-critical events to interested parties**
  - **ia: oaa_Declare(msg(phone, Msg), [])**
  - **pa: oaa_Solve(msg(phone, ringing, []), [inform])**

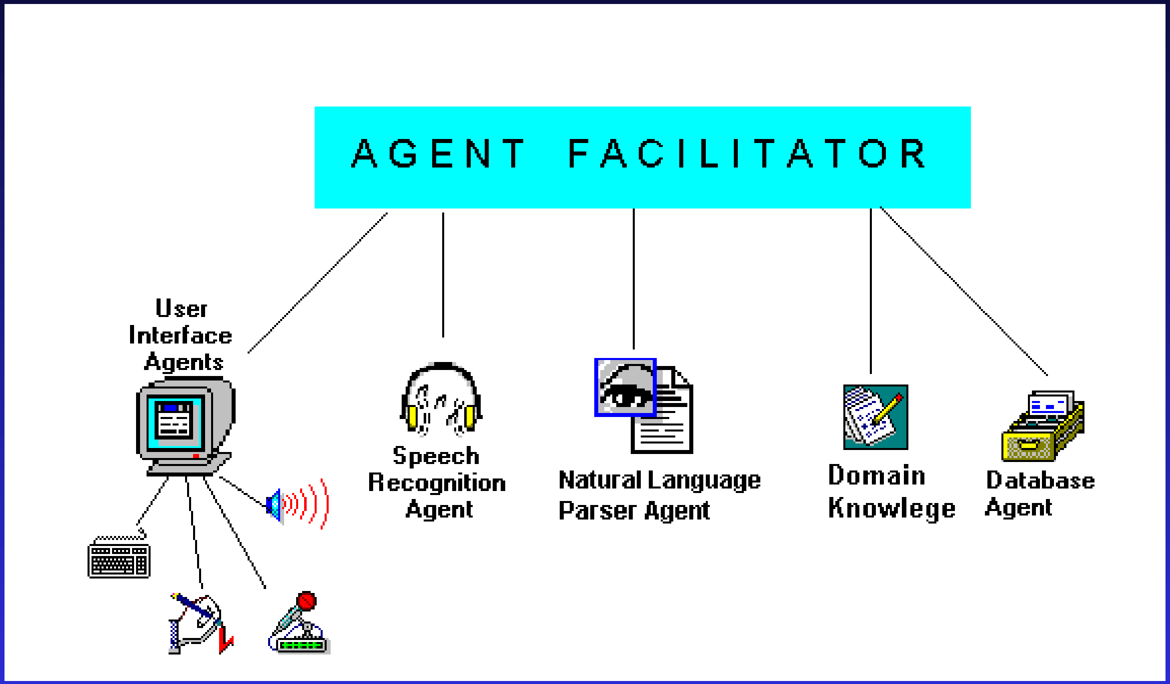


Tips: Domain-Specific vs. Domain Independent Agents

- Move domain-dependent code into separate agent
- Employ hooks and parameters to allow domain-specific tailoring of functionality

- Always ask:
  Domain-specific or domain independent?
  - Phone agent?
  - Office interface?
  - Notify agent?
  - Speech recognition?
  - Natural language?
  - Facilitator?

Tips: Adding Speech & NL



- User Interface responsible for:
  - accepting user input, sending requests, displaying results
  - controlling interactions of Speech and NL
- Complex interpretation processed by external domain agent

Unified Messaging: Problem

- Universal Access: Access to web, email, voicemail, applications (e.g., calendar, database, scheduler) from multiple interfaces (e.g., web browser, desktop, telephone)
- Delegated triggers to monitor information
- Message dissemination across various media (e.g., fax, printer, email, phone, pager)
  - Locating destination target
  - Plan route according to user preferences & resources
  - Media translation as necessary
- Extensible and distributed! Minimize dependencies among component technologies

Main Points

Unified Messaging: Components

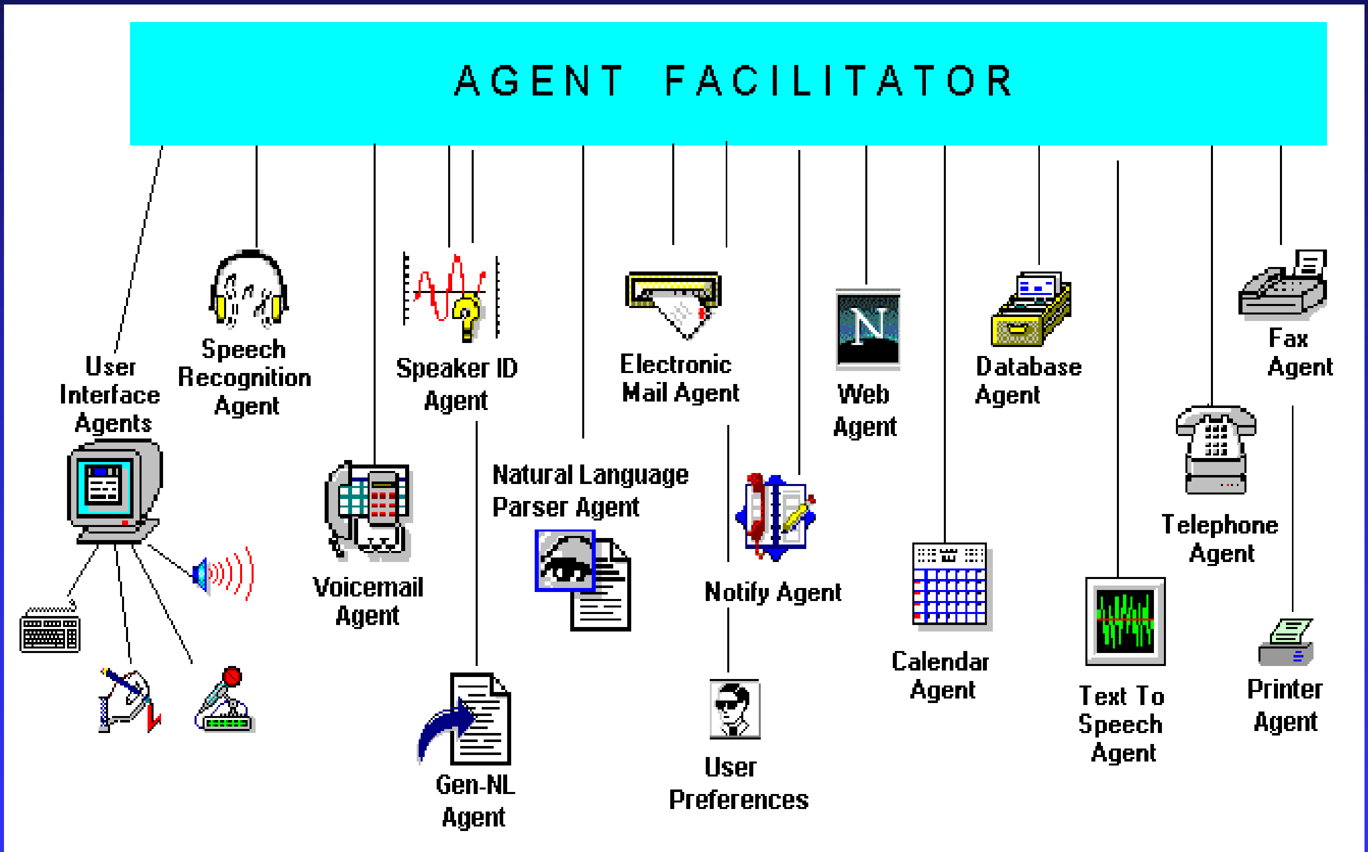
Mobile, adaptable access to distributed services

Integrated Messaging: web, email, voice, fax

Distributed reference resolution and media format translation

Flexible interactions among components

Delegated Triggers



Unified Messaging: Implementation 1/2

- **Universal Access**
  - **Every user interface (including phone) must identify user**
  - **UI's coordinate themselves to ensure only one "primary" interface per user, per utterance**

- **Message Dissemination**
  - **Media agents: distributed reference resolution and translation**
  - **print(Object, Params)**
    - ref(it):
      oaa_Solve(resolve_reference(the, document, Params, NewObj))
    - id(Pointer):
      oaa_Solve(resolve_reference_id_as(id(Pointer), postscript, [], PostScript)
    - print TextObject or PostScript

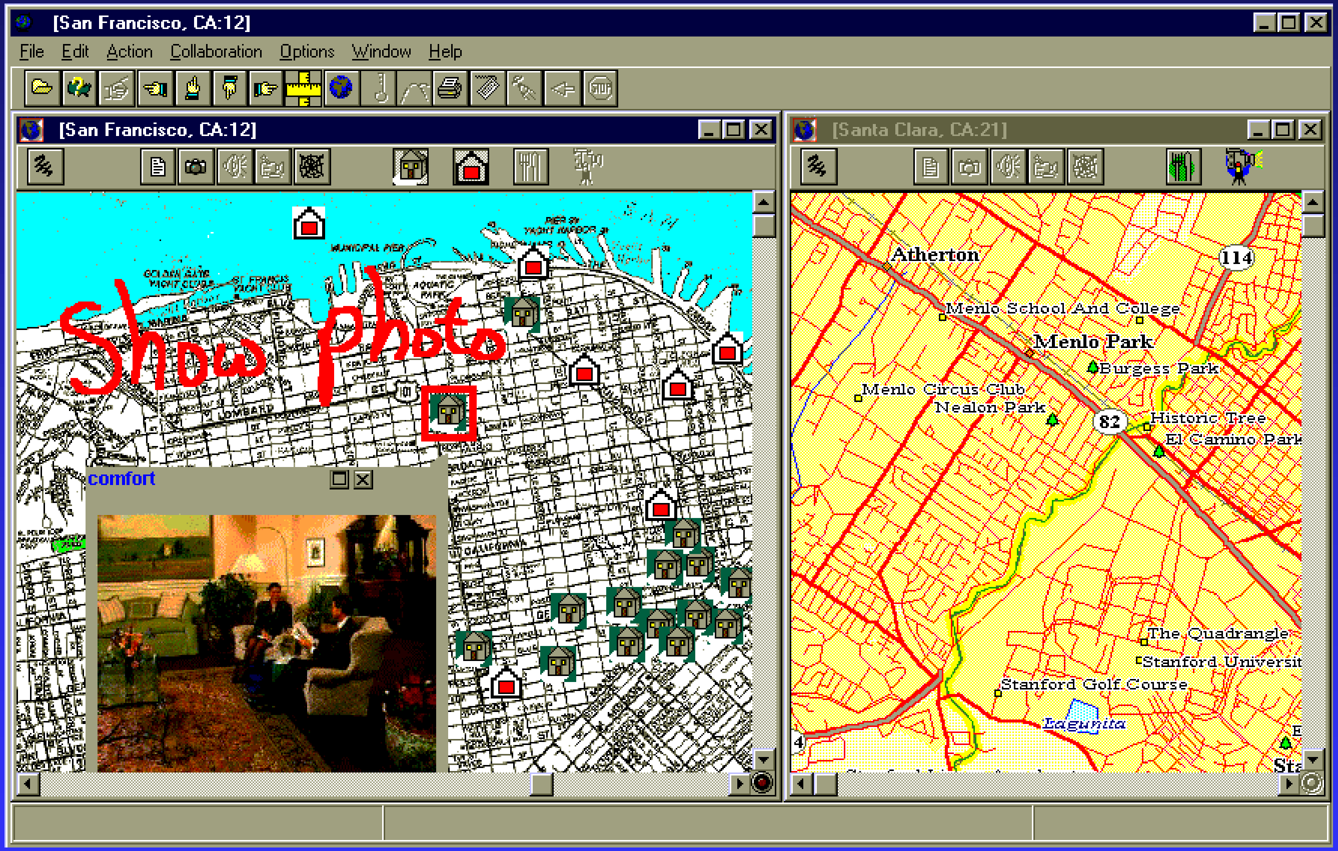
Unified Messaging: Implementation 2/2

- **Adaptable Presentation**
  - **GenNL agent produces simple or structured text-based response for any ICL query**
  - **Reads distributed NL vocabulary definitions in forming simple responses:**
    - **Vocabulary**: noun('telephone number', phone_number, [])
    - **NL -> ICL**: "What is Adam Cheyer's telephone number?"
    - **ICL**: oaa_Solve(phone_number('Adam Cheyer', X),[query(var(X))])
    - **Reponse**: [phone_number('Adam Cheyer', '859-4119')]
    - **GenNL**: "The telephone number of Adam Cheyer is 859-4119."
  - **Structured response: description(list(EltList, AttrList))**
    - title(Title): Title of list, e.g. 'Schedule'
    - elt(Elt): Name of individual element in list, e.g. 'Appointment'
    - intro(Intro): Introduction to be played at start of list, e.g., 'Here is today's schedule for Adam Cheyer'
    - max_len(Max): Num < Max Display All, else Display 1st & iterate

Main Points

Multimodal Maps Application

- Natural interface to distributed (web) data
- Synergistic combination of handwriting, drawing, speech, direct manipulation
- Parallel cooperation and competition among many agents
- Human & Agent collaboration
- Adaptable displays according to user preferences



Multimodal Interfaces using Parallel Distributed Agents

- Competition and cooperation among agents at many levels
  - Pen input: gesture recognizer vs. handwriting recognizer
  - Natural language: multiple NL systems (multilingual, diff. capabilities)
  - Reference Resolution

- Multiple modalities for resolving ambiguities
  - e.g. arrow + "scroll map" vs.
    arrow + "photo of this hotel"

Multimodal Reference Resolution

- Context by object type:
  "show photo of the hotel"
- Deictic:
  "Find distance from here to here", "this one"
- Positional context: Write *"photo?"* on hotel
- Visual context: *"Photo of the [visible] hotel"*
- Database queries:
  "show photo of the hotel in Menlo Park"
- Discourse: *"No, the other one"*
- User disambiguation through prompting:
  "Which hotel?"

Information Broker: Requirements

- Integrate Internet sources with enterprise sources
- Heterogeneity handled transparently
- Structured and “semi-structured” sources
- Flexible access to unreliable information sources
- Easily extensible to new domains
- User and task models used to guide retrievals
- Infrastructure must provide a basis for tools

Information Broker: Functionality

- Mediation
- Retrieval Strategies
- User & Task Models

Mediation

- Transparent access to heterogeneous sources
  - WWW structured and semi-structured sources
  - SQL sources
  - Knowledge bases
  - Multimedia repositories
- Dynamic source registration & schema update
- Query planning across distributed sources
- Queries in broker or source schema
- Domain knowledge used to increase query range
- Built-in normalizations and conversions
- Incomplete & inconsistent information

Retrieval Strategies

- Identification of relevant sources
- Extraction of desired information
  - Imposing structure on semi-structured Web pages
- Local caching of virtual databases
- Sensitivity to time constraints
  - Flexible strategies for web vs. cache retrievals
- Dealing with unreliability and change
  - Cache maintenance
  - Use of alternate sources
  - Tracking and rating of sources

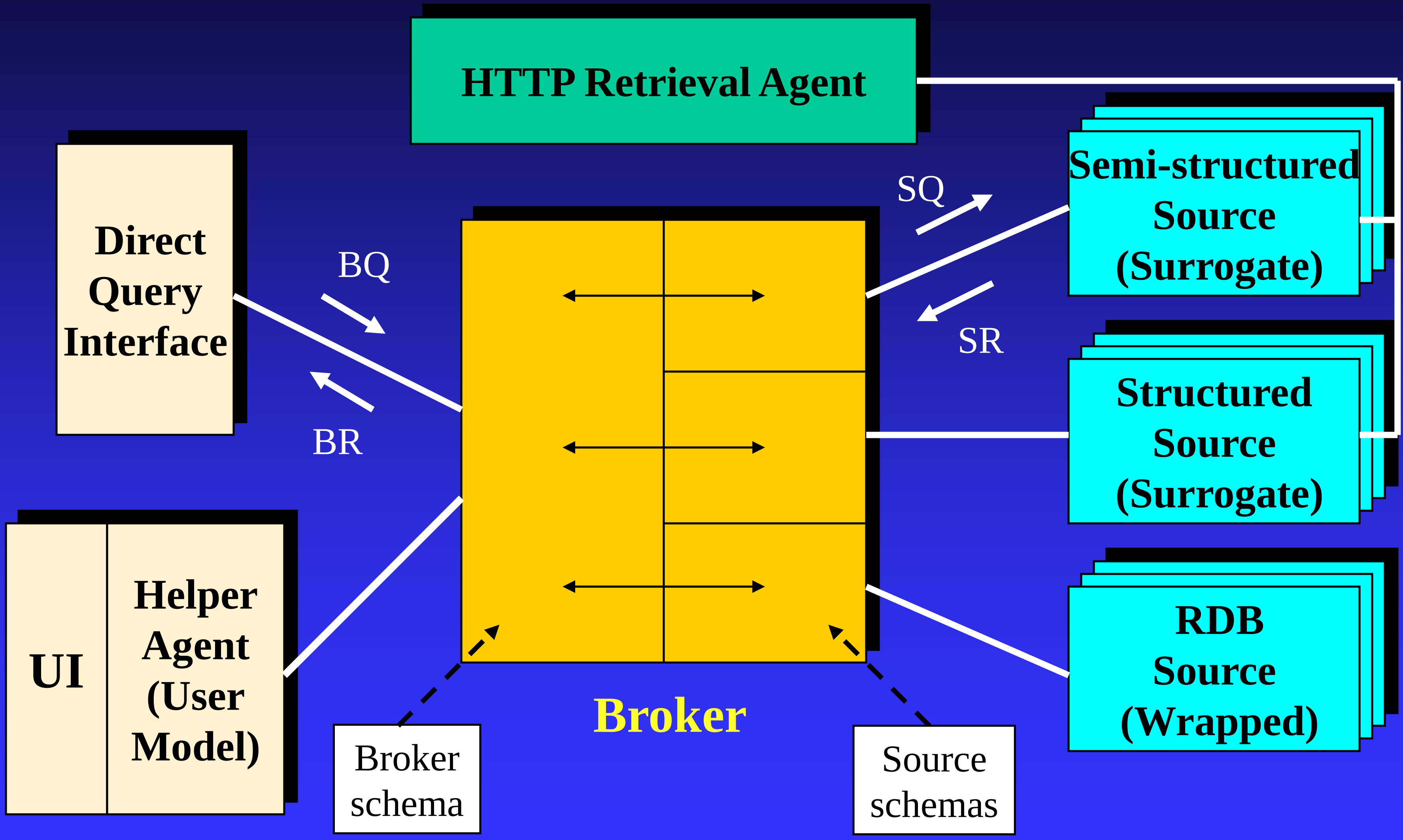
User and Task Modeling

- Representation of salient characteristics of users and tasks
- Mapping from situation to information request
  - What information is needed and when?
  - User and task models used as *constraints*
- Mapping information retrieval to presentation
  - What information does the user want to see?
  - User and task models used as *filters*
- User-friendly knowledge acquisition
- Learning user and task models where feasible

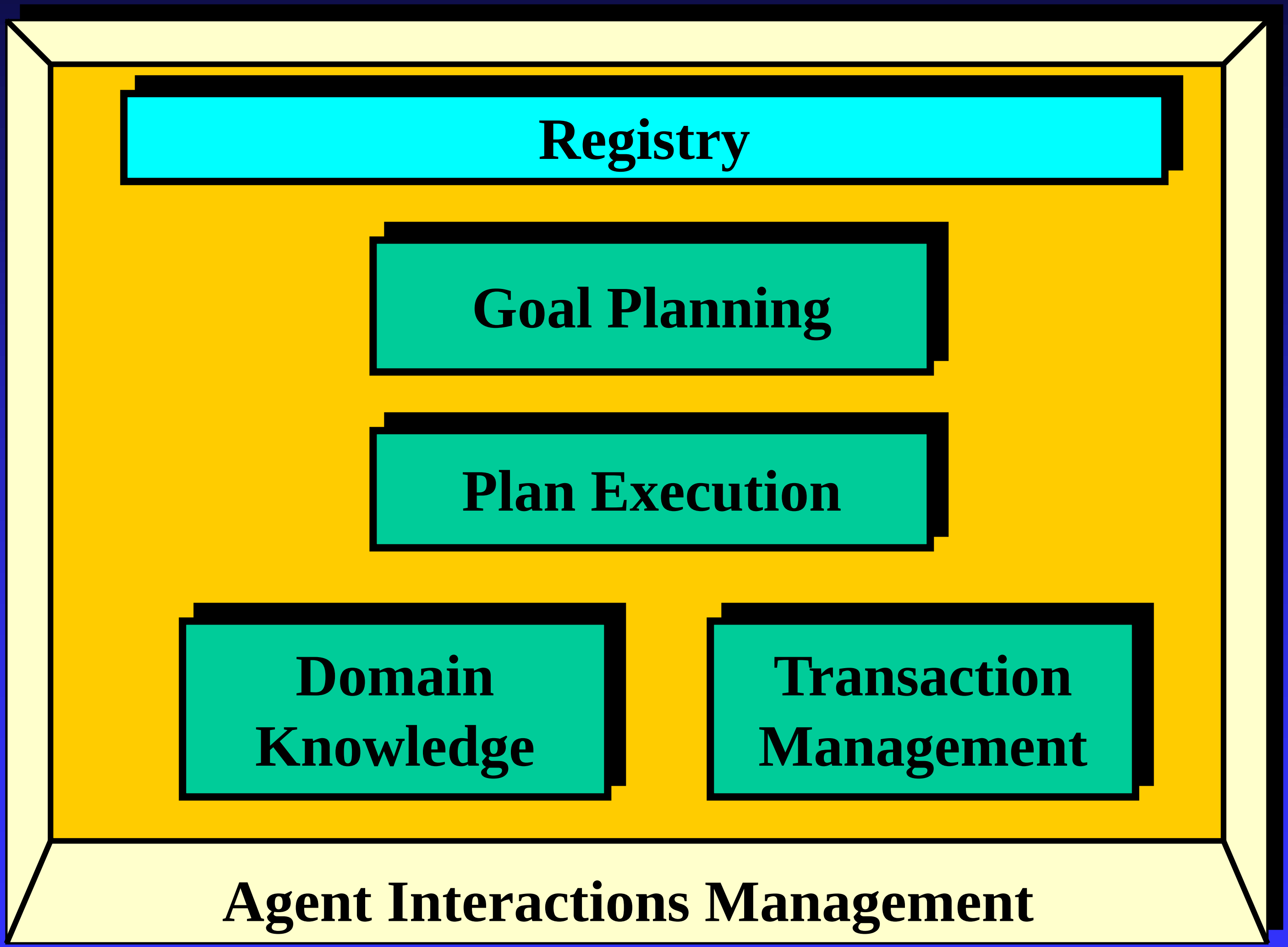
Sample Queries

- **Mediation**
  - "Find all hotels (meeting certain constraints) in San Francisco"
- **Use of domain knowledge**
  - "Find hotels halfway between S. F. and Portland"
- **User modeling**
  - "Apply my preferences" (to the same query)
- **Legacy and Web data source integration**
  - "Show just the hotels for which we get a corporate discount"
    (Accesses WWW sources and employee db)
  - "Find the names and extensions of employees in the AI Center who have written about …"
    (Accesses Harvest index, Bibtex file and employee db)
- **"Persistent" queries**
  - "Notify me of any ad selling a used color inkjet printer"

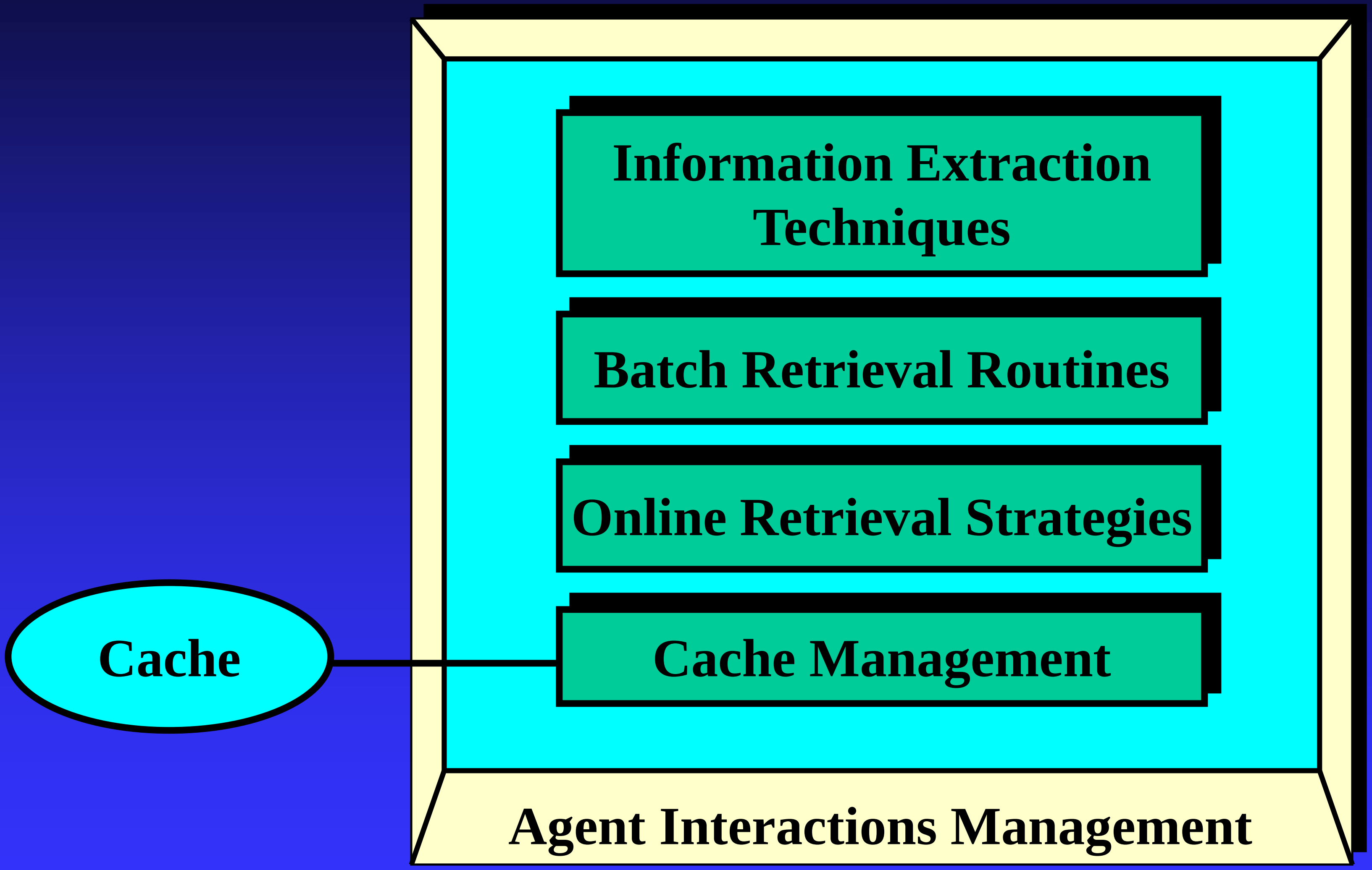
Information Broker: Architecture



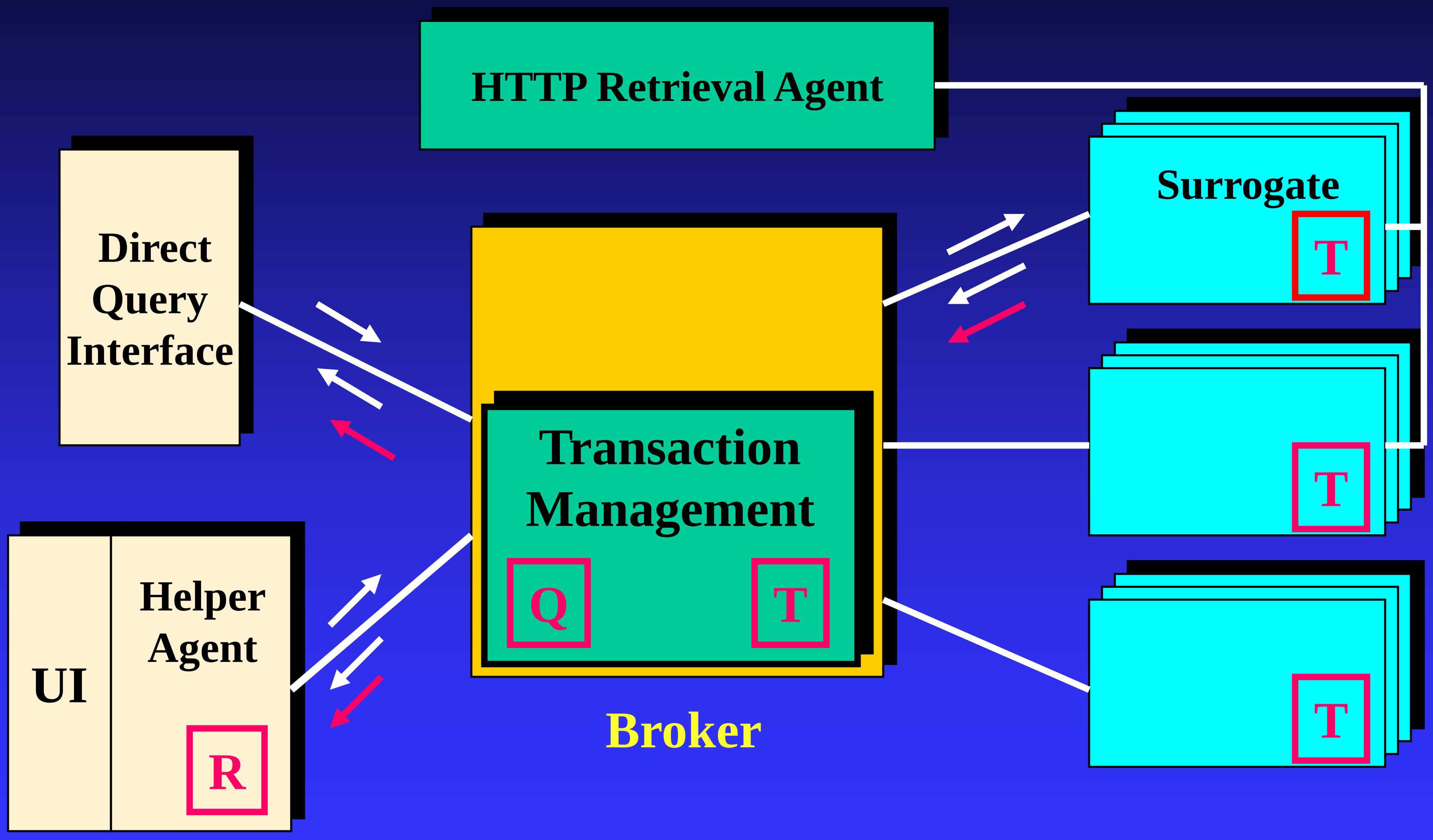
The Broker


Registry
Goal Planning
Plan Execution
Domain Knowledge
Transaction Management
Agent Interactions Management

Surrogates


Information Extraction Techniques
Batch Retrieval Routines
Online Retrieval Strategies
Cache Management
Cache
Agent Interactions Management

Persistent Queries



Useful Features of the Framework

- Tight Integration of Data Capabilities
- Standardized, Visible Content Language
- Extension of Logic Programming Paradigm

Collaboration-ready Data Management

- Store data using OAA Data Management
  - oaa_DbDeclare(icon(Id, X, Y, PictureType), [shareable, callback(icon_change)])
- Separate code which changes data from results, using callback feature
  - NOT:

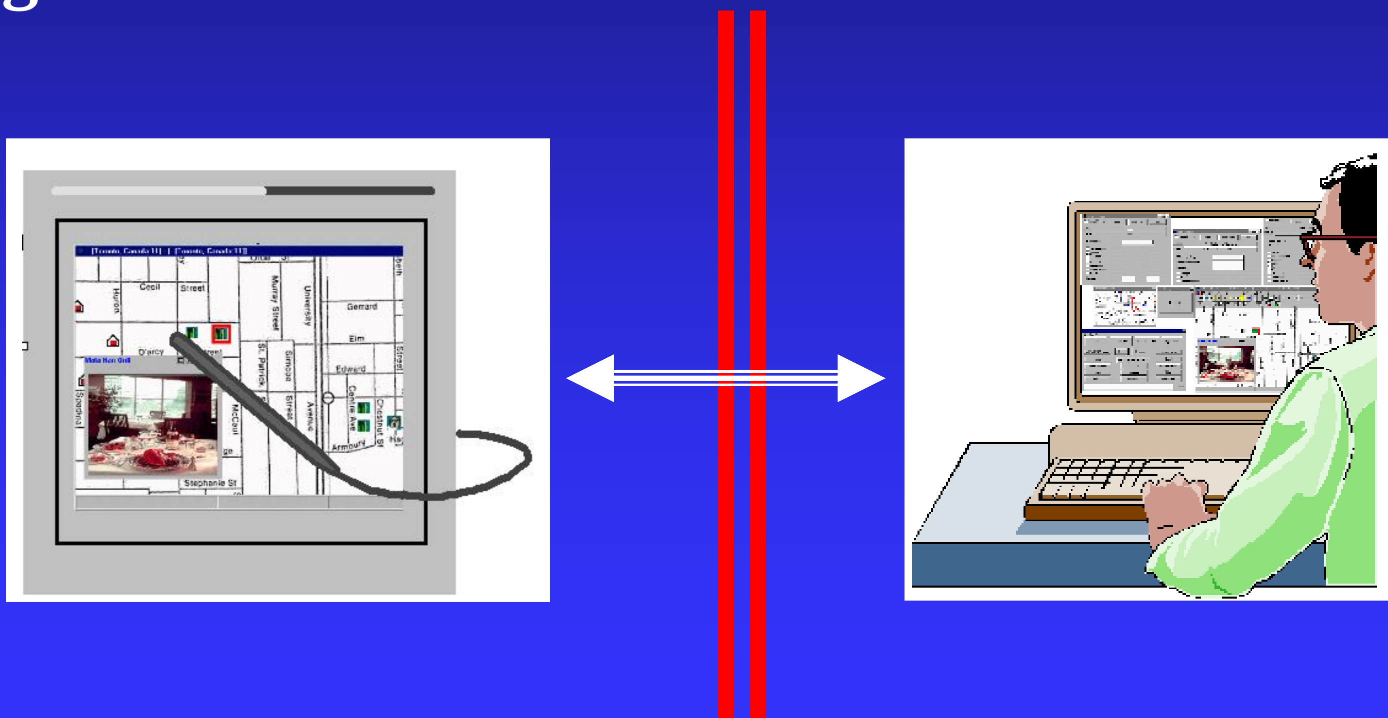
    { oaa_AddData(icon(hilton, 100, 100, hotel), [])
    map_Display(icon(hilton, 100, 100, hotel)) }
  - BUT:

    { oaa_AddData(icon(hilton, 100, 100, hotel), []) }

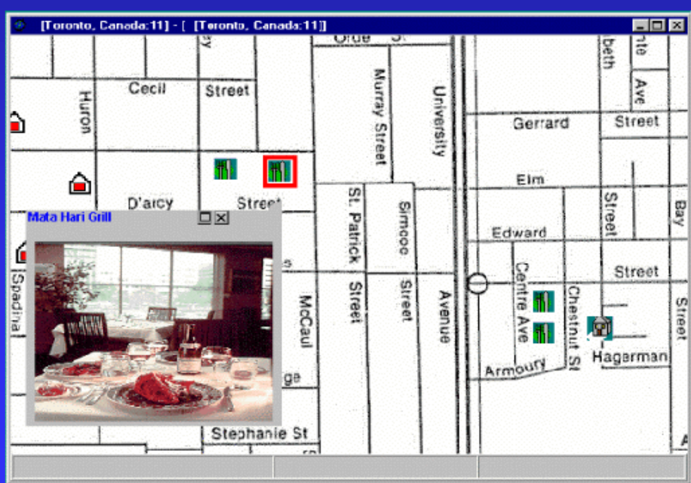
```
icon_change(add, icon(Id, X, Y, Picture)) :-
    map_Display(icon(Id, X, Y, Picture)).
```

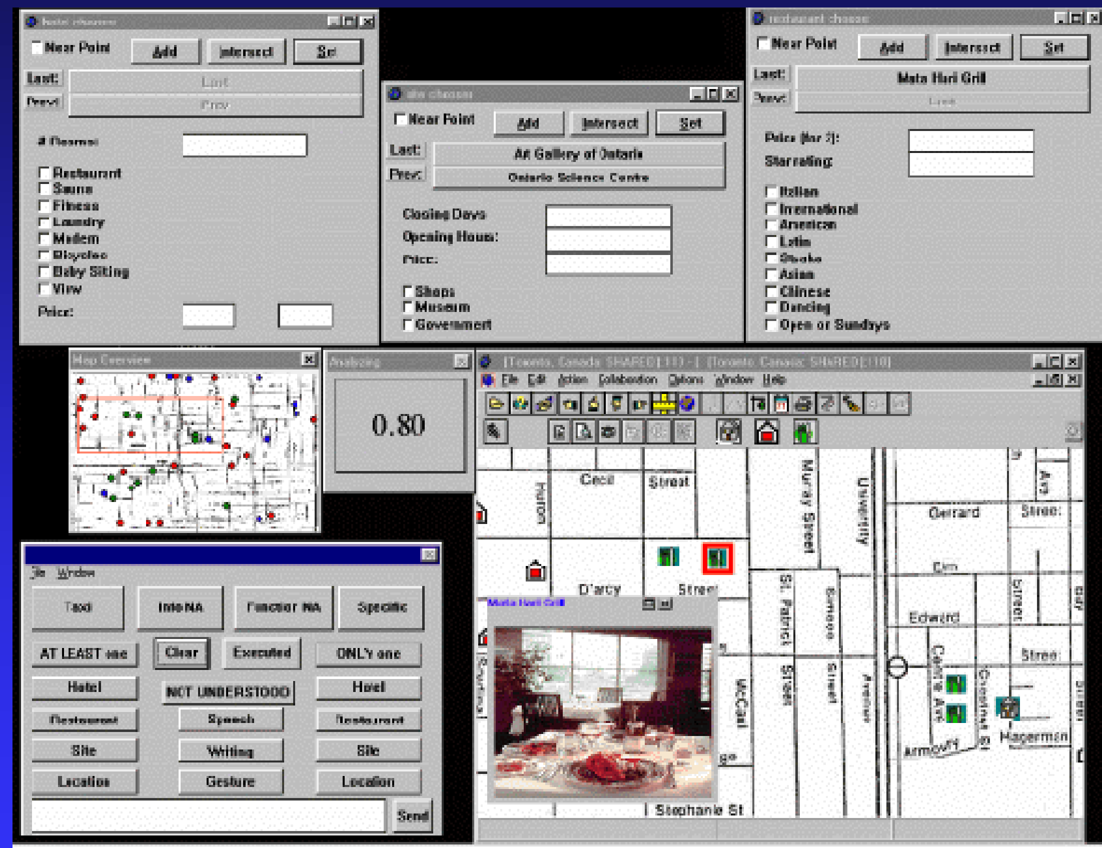
Incremental System Development & Evaluation

- Collaborative Multimodal Map application adapted for Wizard Of Oz (WOZ) experiment to elicit data about coordinated use of language and gesture

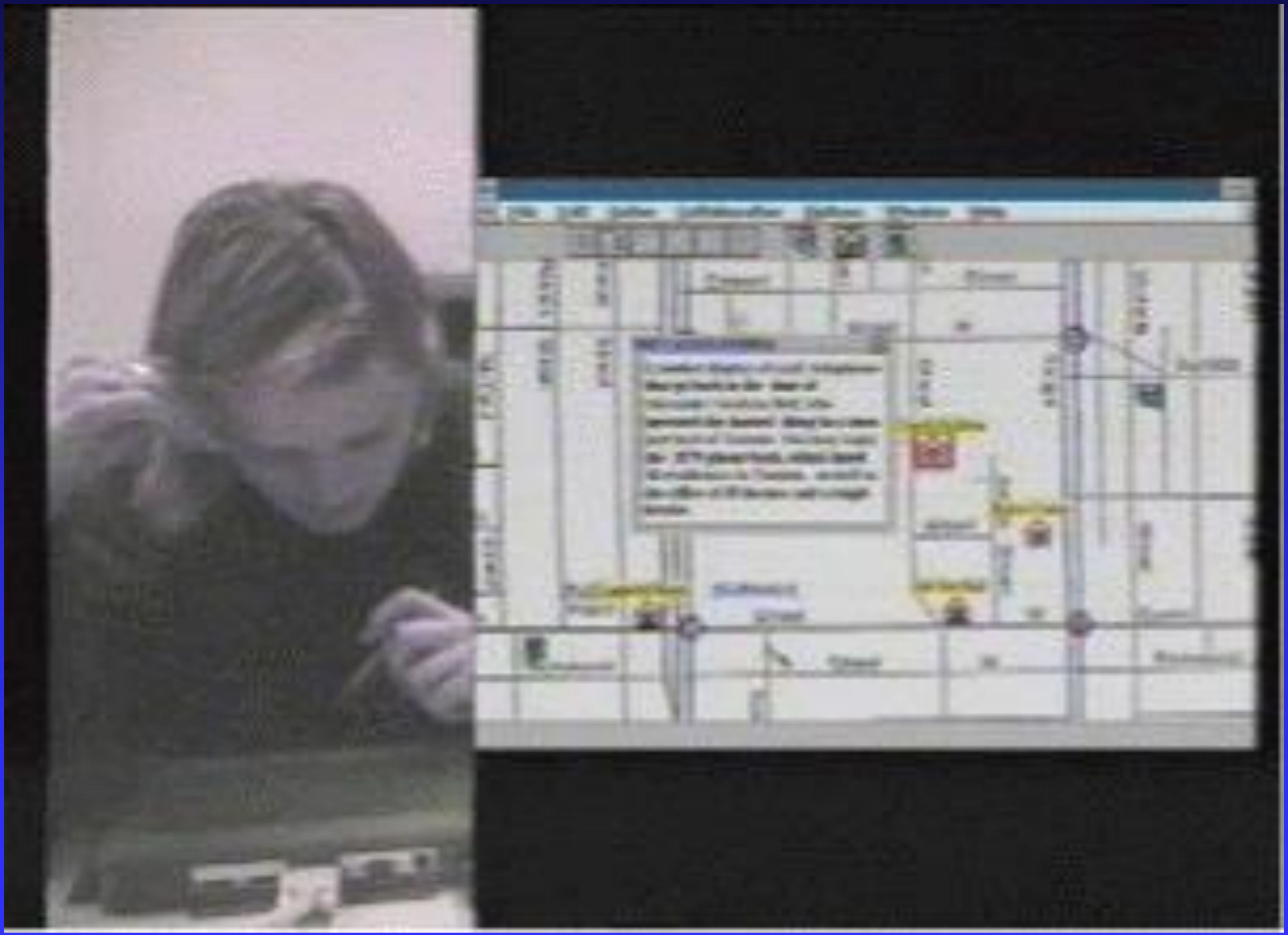


Subject Screen vs. Wizard Screen





Subject Video



Hybrid Wizard Of Oz Experiment

- Naive user free to write, draw, or speak without constraints imposed by current technology
- Wizard must respond quickly and accurately by using existing means, including pen and voice
- Simultaneous evaluation of:
  - Experienced user manipulating real system
  - New user, providing data for future extensions
- Bootstrap effect: continuous loop from data to theory, to system enhancement
- Improvements from data analysis *quantifiable*
- General-purpose approach

Hybrid WOZ: Implementation

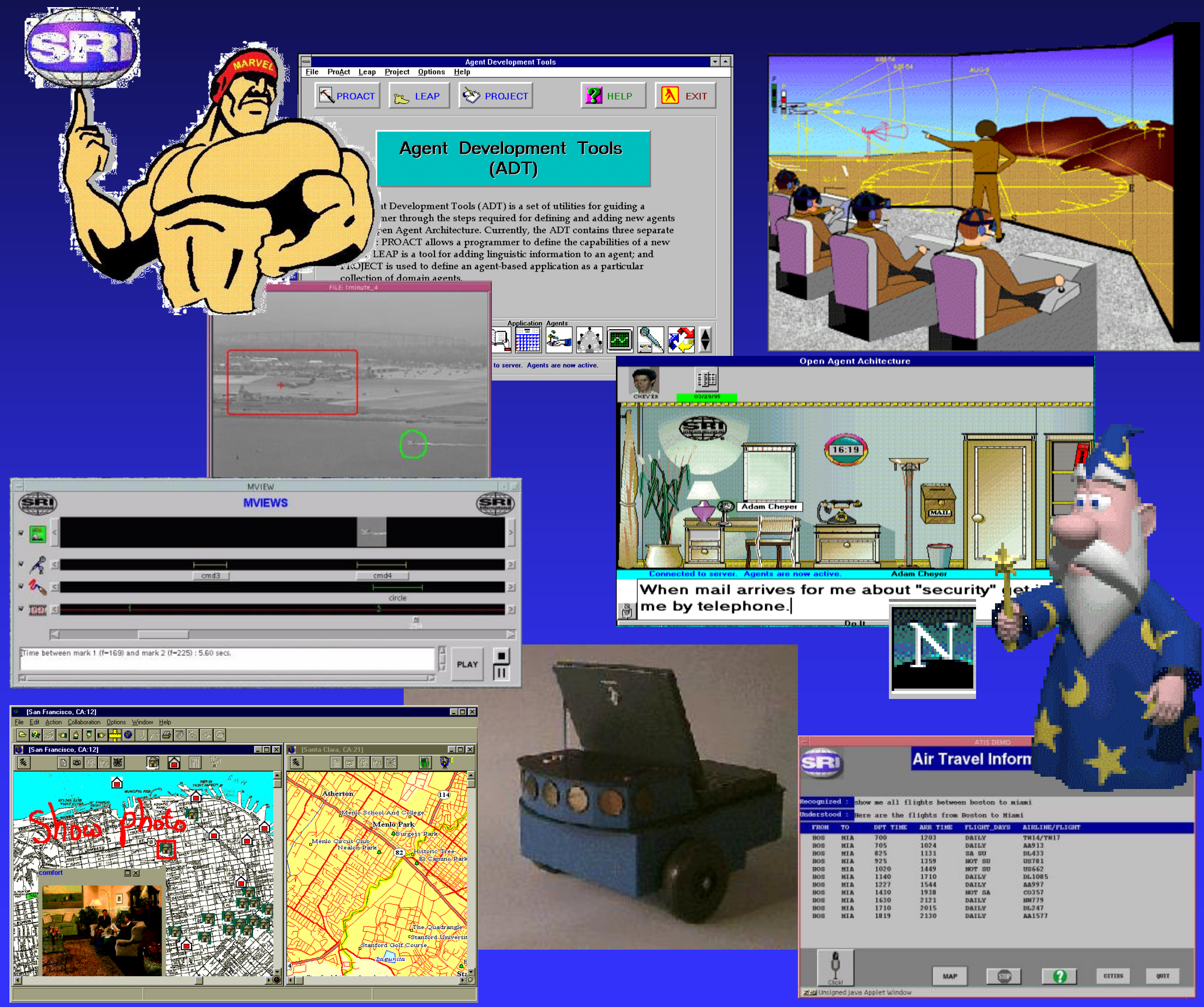
- System logging and playback "for free" using OAA collaboration facilities
- "Subject mode": functional interpretation (mostly) turned off
- Addition of simple *Wizard Feedback* panel (separate agent) for text-to-speech messages (e.g., "Function not available.")

Looking for Killer Apps

- OAA has been used to implement more than 25 systems and prototypes
- Not good for every application, but good for:
  - integrating numerous components which need to cooperate, often across language boundaries
  - supporting media translation
  - distributed reference resolution
  - tasking through adaptable or multimodal user interfaces
  - human/agent collaborative systems & incremental dvpt
  - exploring direct manipulation/task delegation tradeoffs

OAA-based Applications

1. Automated Office
2. Unified Messaging
3. Multimodal Maps
4. CommandTalk
5. ATIS-Web
6. Spoken Dialog Summarization
7. Agent Development Tools
8. InfoBroker
9. Rental Finder
10. InfoWiz Kiosk
11. Multi-Robot Control
12. MVIEWS Video Tools
13. MARVEL
14. SOLVIT
15. Surgical Training
16. Instant Collaboration
17. Crisis Response
18. WebGrader
19. Speech Translation
20-25+ ...



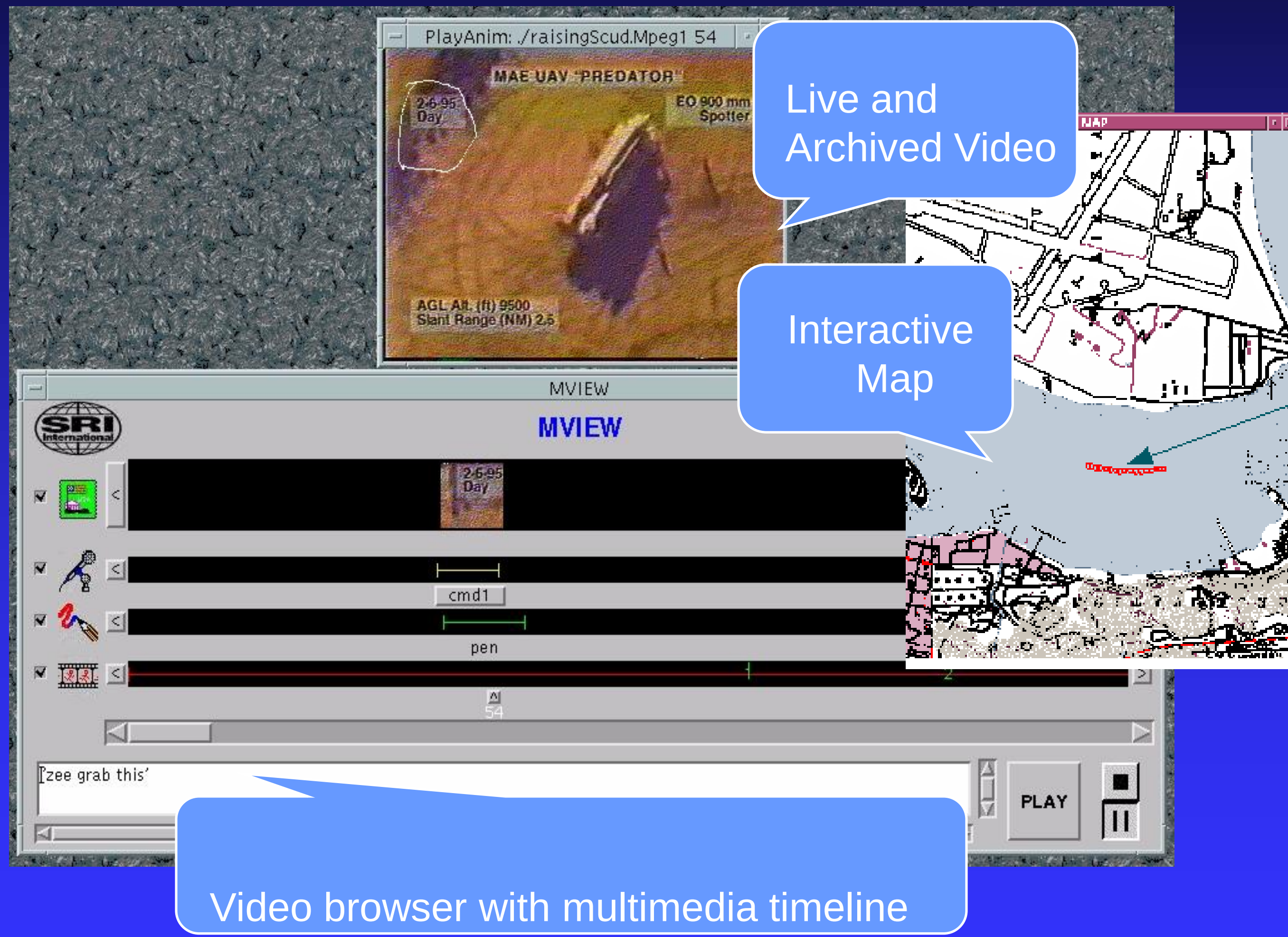
Main Points MVIEWS Application

Multimodal annotation of video using speech & pen

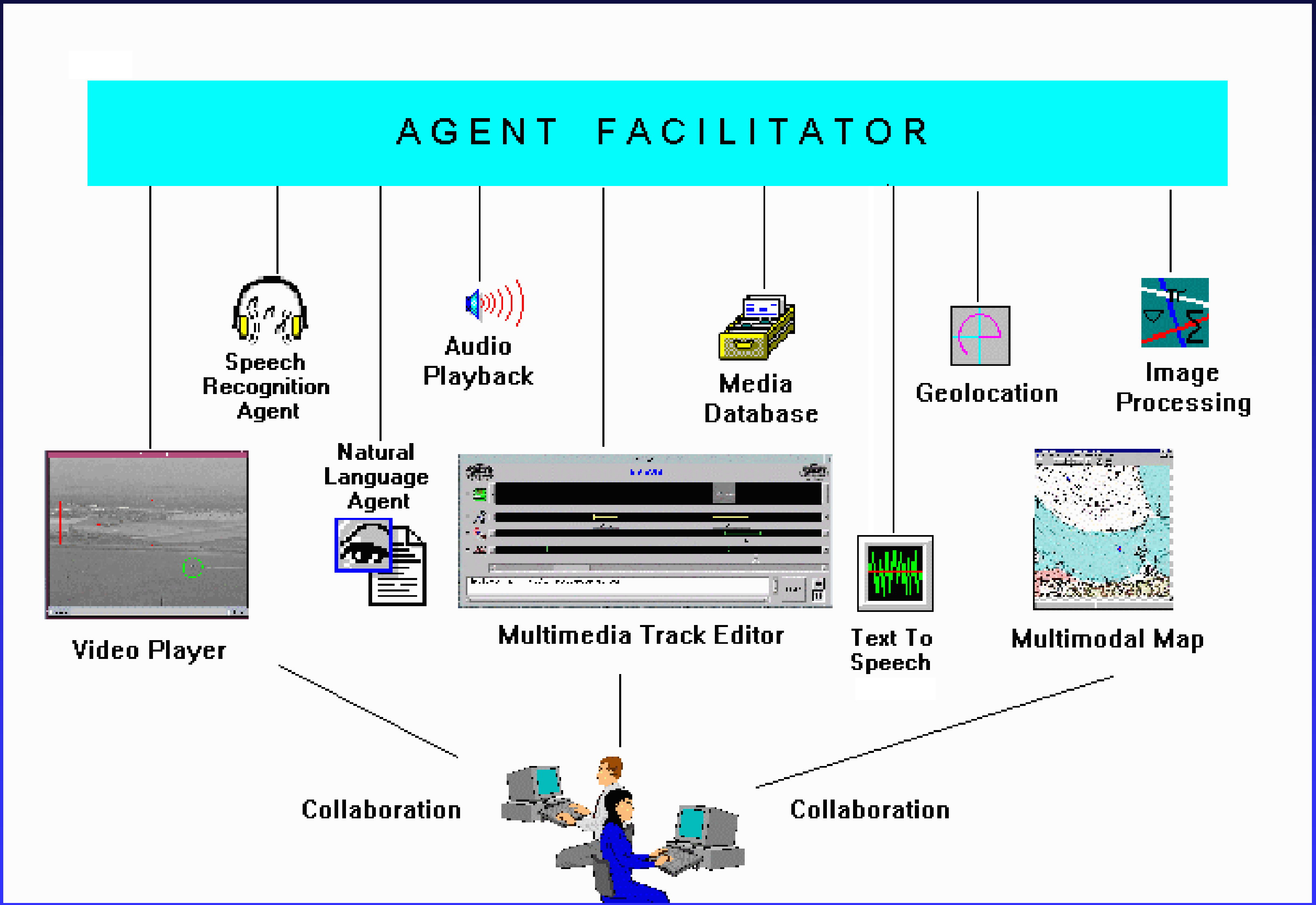
Automated detection, tracking, and geolocation of moving objects

Search and replay of videos indexed by multimodal and auxilliary data

Applications: multi-sensor surveillance, Predator UAV, Olympic bombing



MVIEWS Architecture



Main Points

InfoWiz Kiosk

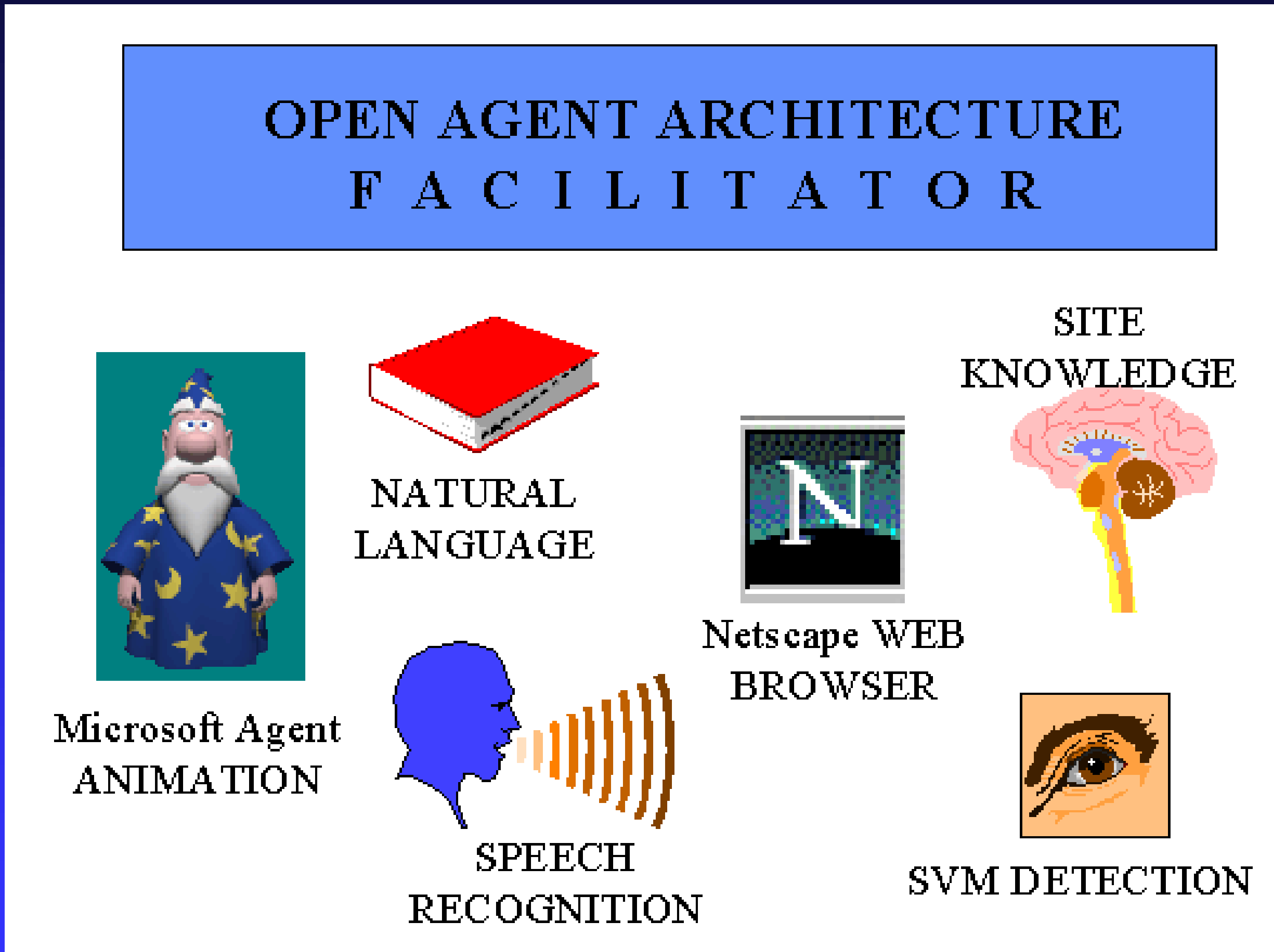
An information kiosk with an animated wizard who :

answers questions, gives tours, and helps navigate the information space

OAA integrates SRI's speech recognition, NL, dialogue, and knowledge representation with Microsoft Agent graphics and Netscape's webbrowser

Soon in SRI 's lobby

InfoWiz Kiosk Architecture



Multi-Robot Control

Monitoring

Maps, video, status

Global or individual views

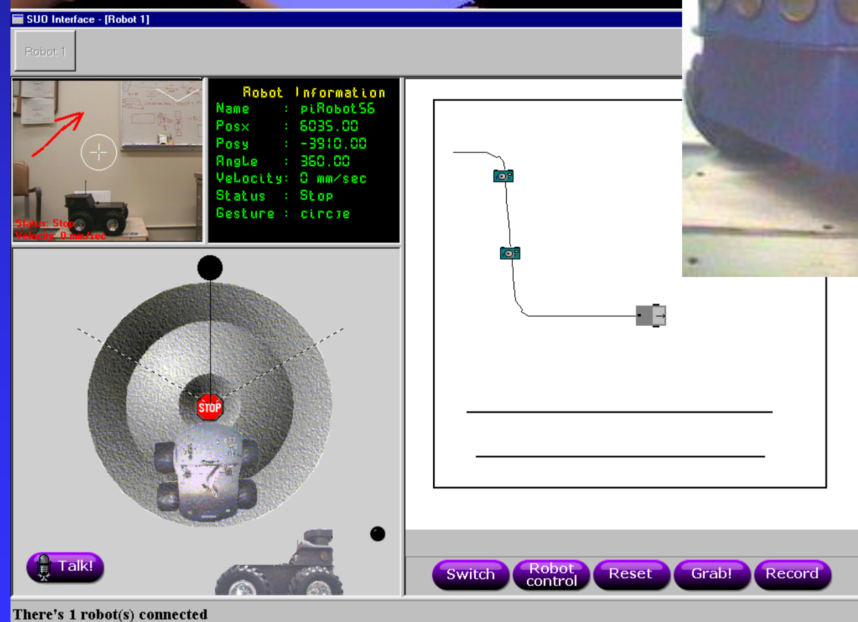
Configurable displays

Tasking

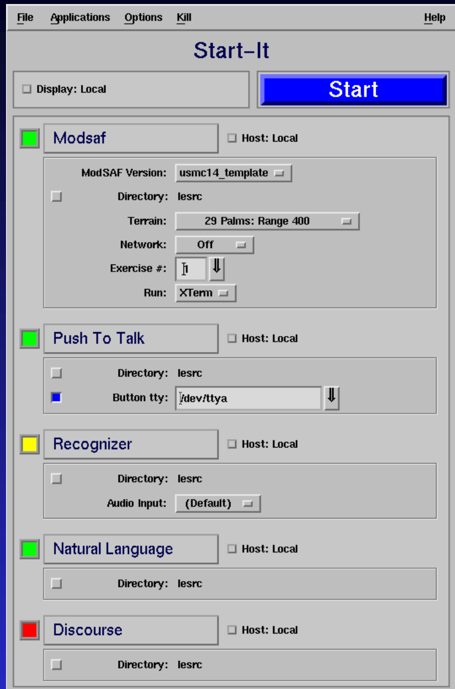
Directed camera & robot control

Delegated tasking through speech & gesture

Concept Design



Agent Development Tools



Tools are implemented themselves in OAA

Guide user through process of creating an agent:

- Definition of capabilities
- Documentation management (publication on Web)
- Code generation of agent template
- Definition of NL vocabulary
- Update NL & speech recognition systems
- Assembly of multiagent projects

Runtime tool for launching and monitoring agent communities

Concluding Remarks

- Many Varieties of
  - Agents
  - Agent-based Systems
  - Agent Frameworks
- Useful Features of Agent Frameworks
- Important Design Choices
  - Strategies for Interoperation & Coordination
  - Managing and Sharing Data
  - User Interface Functionality
  - Framework

Exhibit G

FIPA report

Adam Cheyer (*cheyer@ai.sri.com*)
Mon, 01 Jul 1996 18:40:32 -0700

- **Messages sorted by:** [date][thread][subject][author]
- **Next message:** John Dowding: "dynamic language models"
- **Previous message:** David Martin: "setup.pl in common directory"

Well, I'm back from the 2nd FIPA (Foundation for Intelligent Physical Agents) meeting in Yorktown, NY. FIPA is an effort by the instigator of the MPEG consortium to standardize agent technology. So here's the report...

In general, the workshop started out quite disorganized, basically because so many different points of view were represented: some people were interested in robots (about 10% only), some about multimedia diffusion (video on demand) and learning user's preferences to select appropriate content, some interested in mobile, general magic style agents, and others were more in the distributed software agents area.
Since no one clear vision was presented of what an agent is, much discussion was spent on what would be standardized and what wouldn't.

The second day was a little more efficient, as questions were prohibited during the presentations. We were able to see the broad spectrum of requirements for all the different participants. There were many very interesting presentations condensed into a very short time. We will receive (in a few weeks) slides from all the presentations; I can send you copies if you like.

Of the presentations I saw, my favorites were :

Donald Steiner, Siemens : their MECCA architecture shares many qualities with OAA. They are applying the technology to the domains of office assistant (very similar to OAA!) and to trip planning.

Albert D. Baker (University of Cincinnati): real world domain using a broadcast mechanism to make 1000 (!) agents interact.

Nader Azarmi (BT Labs): very interesting negotiation-capable agents applied to the domain of business process control.

For my brief presentation, I tried to stress the following points:

1. In my opinion, it is important for FIPA to carefully define what is an agent and how this differs from other technologies such as objects. What are the requirements of an agent architecture vs. a distributed object architecture? Can't we reuse many common notions (such as security) that standards like CORBA already address?

2. At SRI, we have one possible definition of what an agent should be, focusing strongly on distributed, delegated "intelligence" and control, and on a high-level communication language.

3. We've implemented a number of applications within the framework (probably more than any other group I saw at FIPA) and with this experience, we've learned some valuable lessons, one being the utility of providing hooks for extensible tools to simplify interfacing with and using agents.

In the third day, some progress was made in defining a list of requirements, target applications, and standardisation items to be considered. "Results" can be seen on the FIPA homepage, at http://www.cselt.stet.it/ufv/leonardo/fipa.htm.

I think that given all the different viewpoints, FIPA seems to be moving towards creating some sort of standard that will primarily reflect a distributed software agents perspective. Everything will become much more concrete during the next meeting in October, which is being held in Japan: at this time, a call for proposals will be issued. I think that, if pushed by some tangible applications which focus the vision a little, FIPA might actually arrive at a defining a useful set of standards that will allow agent interoperation on a larger scale than currently available. The intent is to have the first FIPA standards defined by the end of 1997...

One question is, what is the role of OAA in all this? I believe that if we choose to, we are in a strong position to influence much of what FIPA produces as a standard. I think that perhaps with the Siemens and BT Labs, we really have the closest overall framework and approach for what FIPA is trying to accomplish. And we are the only group who is trying to use their framework to implement the complete range of agent applications, from cooperating robots, to intelligent avatars, to multimodal and multimedia interfaces, to information retrieval and management. The release of a distributable version of OAA-lite should only enhance our bargaining power.

So, that's the report. If you have questions, feel free to ask...

Adam.

- **Next message:** John Dowding: "dynamic language models"
- **Previous message:** David Martin: "setup.pl in common directory"

Exhibit H

OAA-USERS mailing list

Doug Moran (*(no email)*)
Fri, 21 Jun 1996 15:35:24 -0700 (PDT)

- **Messages sorted by:** [date][thread][subject][author]
- **Next message:** David Martin: "bug: srx agent on standalone machine"

All:

We have created a mailing list for discussion among the users of OAA.
The purpose of this list is to allow users to share experience, ask
questions of the larger group, and make suggestions for enhancements
that other users can comment on.

The mailing address of this list is:
oaa-users@ai.sri.com
And an HTML-based archive of messages is accessible at:
<A HREF="http://www.ai.sri.com/~oaa/distribution/MailArchive/oaa-users/index.html">OAA Users archive</A>

Membership:
This list will include
1. the OAA project staff
2. SRI personnel using OAA on other projects
3. outside users of OAA.

Note: messages to this list will be seen by all these other people, not
just the OAA project staff.

To send a message to just the OAA project staff (that other users will not
see), please direct it to the appropriate individuals.

** PLEASE respond if you wish to be included on this list***
If there are colleagues that should be included, please send me a note
and I will add them.
I would also appreciate a response if you do not wish to be included
so that I know that all of you have received this message.

Possible reason for not being on the list:
The sendmail "expn" command shows the membership of this list, and you

might not want to be listed.

Even if you are not on the mailing list, you can check the archives
(the archives are currently in the private portion of the OAA web site
to which SRI personnel and designated SRI clients have access).

-- Doug Moran

- **Next message:** David Martin: "bug: srx agent on standalone machine"

Exhibit I

bug: srx agent on standalone machine

David Martin (*(no email)*)
Thu, 27 Jun 1996 16:37:05 -0700 (PDT)

- **Messages sorted by:** [date][thread][subject][author]
- **Next message:** David Martin: "setup.pl in common directory"
- **Previous message:** Doug Moran: "OAA-USERS mailing list"

oaa-users -

Recently 2 different OAA user groups uncovered a problem with the "srx" (Speech Recognition) agent, which occurs ONLY when an agent system is being run on a standalone machine (that is, detached from the network).

The problem is that srx is not able to establish communications with the Nuance (or Corona) recognition server. The symptom is an error message from srx that says:
NUANCE_SERVER_NOT_ACCESSIBLE
or
CORONA_SERVER_NOT_ACCESSIBLE

The problem turned out to be quite hard to diagnose, but it now appears that it's due to a nonstandard C library (libc.a) with which srx was linked. This is a nonstandard library that was created for use at SRI, and which changes the normal behavior in resolving host names.

This same nonstandard library could conceivably affect other agents compiled at SRI, but as far as we know, only srx has exhibited a problem.

There appears to be a simple workaround (although it has not yet been tested very extensively):

Add the following line to the file
/etc/resolv.conf
on the standalone machine:

order local,bind

On our machines here at SRI, it is the first non-commented line, so it might be wise to place it in that position.

- Dave Martin

- **Next message:** David Martin: "setup.pl in common directory"
- **Previous message:** Doug Moran: "OAA-USERS mailing list"

Exhibit J

OAA-USERS Mailing List Archive by thread

- **Most recent messages**
- **Messages sorted by:** [date][subject][author]
- **Other mail archives**

Starting: *Fri 21 Jun 1996 - 00:00:-30276 PDT*
Ending: *Thu 26 Aug 1999 - 17:01:06 PDT*
Messages: 40

- **OAA-USERS mailing list** *Doug Moran*
- **bug: srx agent on standalone machine** *David Martin*
- **setup.pl in common directory** *David Martin*
- **FIPA report** *Adam Cheyer*
- **dynamic language models** *John Dowding*
  - **Re: dynamic language models** *David Martin*
- **problem with passing floats in OAA** *John Dowding*
- **passing floating point numbers through library(tcp)** *John Dowding*
- **john makes a mistake** *John Dowding*
- **StartIt: onready, ondisconnect** *Adam Cheyer*
- **Alert: XWindows agents** *Adam Cheyer*
- **Release Version** *Adam Cheyer*
- **Trigger Changes** *Adam Cheyer*
- **New event scheme** *Adam Cheyer*
  - **Re: New event scheme** *Luc JULIA*
  - **Re: New event scheme** *David Martin*
  - **Re: New event scheme** *Adam Cheyer*
- **Proposal: New Data management** *Adam Cheyer*
  - **Re: Proposal: New Data management** *David Martin*
- **PC News** *Adam Cheyer*
- **C Library: structured message proposal** *Adam Cheyer*
  - **Re: C Library: structured message proposal** *Victor S. Abrash*
- **Re: C Library: structured message proposal** *David Martin*
- **Re[2]: C Library: structured message proposal** *Martin Fong*
- **Re[3]: C Library: structured message proposal** *Keith Skinner*
- **oaa-users mailing list** *Adam Cheyer*
- **OAA Mailing List** *The Emminizer's*
- **Strings in the OAA C library** *Adam Cheyer*
- **Re: Strings in the OAA C library** *Ralph Becket*

- **Inquiry concerning NLP API** *Sharon Settnek*
  - **Re: Inquiry concerning NLP API** *Adam Cheyer*
- **Loopy Triggers** *Stuart Lowry*
  - **Re: Loopy Triggers** *Adam Cheyer*
- **writeBB <-> readBB question** *Stuart Lowry*
  - **Re: writeBB <-> readBB question** *Adam Cheyer*
- **New facilitator prompting for port** *John Dowding*
- **OAA library for Perl** *John Dowding*
  - **Re: OAA library for Perl** *Luc JULIA*
- **Large data sets** *Stuart Lowry*
  - **Re: Large data sets** *Adam Cheyer*

Last message date: *Thu 26 Aug 1999 - 17:01:06 PDT*
Archived on: *Thu Aug 26 1999 - 17:01:12 PDT*

- **Messages sorted by:** [date][subject][author]
- **Other mail archives**

This archive was generated by hypermail 1.02.

Exhibit K

iui 98

1998
international
conference
on
Intelligent User Interfaces

acm PRESS

san francisco, california
january 6-9, 1998

iui 98

1998
international
conference
on
Intelligent User Interfaces

acm PRESS

san francisco, california
january 6-9, 1998

The Association for Computing Machinery
1515 Broadway
New York New York 10036

ACM ISBN: 0-89791-955-6

Additional copies may be ordered prepaid from:

ACM Order Department
PO Box 12114
Church Street Station
New York, NY 10257

Phone: 1-800-342-6626
(US and Canada)
+1-212-626-0500
(all other countries)
Fax: +1-212-944-1318
E-mail: acmpubs@acm.org

ACM Order Number: 608980

Printed in the USA

TABLE OF CONTENTS

TUTORIALS

PLENARY ADDRESS I

PAPERS I: Animated Agents

PANEL I: Speech Research: Near and Not-So-Near Results and What They Might Mean for IUI

PAPERS II: Integration

PAPERS III: Tasks and Usage

PANEL II: Affect and Emotion in the User Interface

PLENARY ADDRESS II

PAPERS IV: Demonstrational Interfaces

PAPERS V: Intelligent Database Interfaces

PAPERS VI: Adaptation and Critiquing

PAPERS VII: Evaluation

MVIEWS: Multimodal Tools for the Video Analyst

Adam Cheyer
Artificial Intelligence Center
SRI International
333 Ravenswood Avenue
Menlo Park, CA 94025
+1 415 859 4119
cheyer@ai.sri.com

Luc Julia
STAR Laboratory
SRI International
333 Ravenswood Avenue
Menlo Park, CA 94025
+1 415 859 4269
julia@speech.sri.com

ABSTRACT

Full-motion video has inherent advantages over still imagery for characterizing events and movement. Military and intelligence analysts currently view live video imagery from airborne and ground-based video platforms, but few tools exist for efficient exploitation of the video and its accompanying metadata. In pursuit of this goal, SRI has developed MVIEWS, a system for annotating, indexing, extracting, and disseminating information from video streams for surveillance and intelligence applications. MVIEWS is implemented within the Open Agent Architecture, a distributed multiagent framework that enables rapid integration of component technologies; for MVIEWS, these technologies include pen and voice recognition and interpretation, image processing and object tracking, geo-referenced interactive maps, multimedia databases, and human collaborative tools.

Keywords

Multimodal pen and voice user interfaces, image processing and object tracking, video analysis and annotation, agent architecture.

INTRODUCTION

Although sophisticated tools are now starting to appear that assist an image analyst in manipulating still photos, few systems exist to help an operator efficiently exploit full-motion video. Given video's inherent advantages for characterizing events and movement in a scene, the role of video analysis is taking on increased importance in military, intelligence, and surveillance domains. By considering how video can be best exploited in these contexts, we realize that video analysis poses new challenges and opportunities:

IUI 98 San Francisco CA USA

- At the 1996 Atlanta Olympics, after a bomb went off inside of Olympic Park, investigators had to deal with the task of thoroughly searching for clues within some 600 amateur videos related to the incident. Had there been better tools available for indexing, time stamping, annotating, classifying and cross-referencing the videos, this process would have been much more manageable.
- The U.S. military, as part of peacekeeping and intelligence missions, routinely sends unmanned Predator airplanes over target sites of interest. While pilots remotely guide the airplane over the terrain, a second team of analysts is responsible for extracting information of relevance from the video and associated metadata. Although all results of these missions are recorded on videocassette, there is currently no automated method for querying this data repository at a later date. A searchable index would help ensure that this resource is not wasted, and providing the ability to replay audio and written annotations would help the analysts reviewing a video to quickly establish context for what they are seeing.
- In many surveillance or security-related tasks, a single operator is responsible for monitoring the output of many cameras distributed throughout the site. Object detection and tracking, in conjunction with automated alerts and sensor management, can augment the operator's ability to efficiently comprehend and fuse numerous information streams.

In this paper, we present a demonstration system called MVIEWS, for Multimodal Video Imagery Exploitation WorkStation, that attempts to address some of these challenges by bringing together multiple commercial and research technologies into a single toolset. Although each technology is interesting by itself, it is the *integrated use* of these capabilities that can greatly enhance the effectiveness of a video analyst.

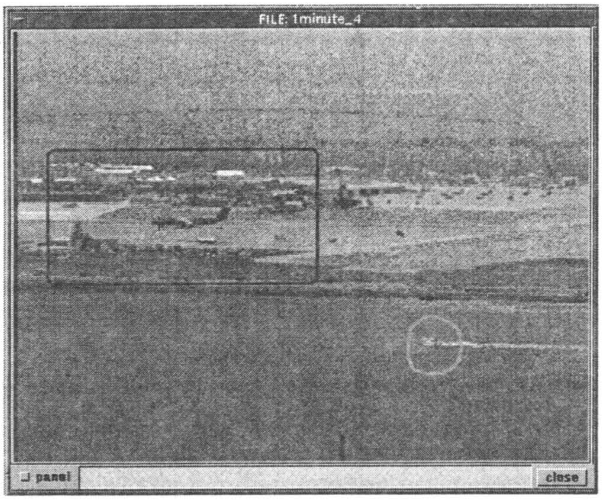


Figure 1. Video Player. Moving objects in surveillance regions are tracked (plane). Multimodal commands can target specific objects (boat).

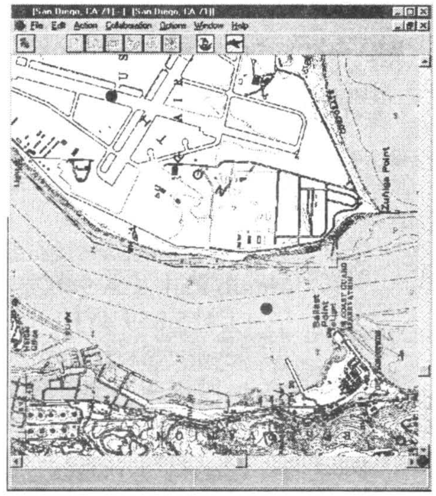


Figure 2. Multimodal Map. Objects tracked in a video are simultaneously displayed on a geo-referenced map.

SYSTEM DESCRIPTION AND DESIGN

Design Approach

The original design for MVIEWS was based on a vision and on a set of guiding principles. The vision can be described as follows.

Imagine a single operator at a terminal, exploiting ten ground and aerial video sensors. Software agents are providing real-time object detection, alerts, queuing, tracking, and information fusion. The operator uses only voice commands and an electronic pen to control the workstation and sensors, to add multimodal annotations to the video streams, and to collaborate with operations and intelligence specialists at remote locations. As a situation develops, agents and humans work together on assessment and characterization, while documenting the process through semiautomatically generated reports.

When setting out in pursuit of this vision, we tried to also keep in mind a few frequently overlooked design criteria:

- People are indispensable. They are good at processing complex visual problems, but they are subject to fatigue and boredom. Automation is an aid, not a replacement.
- User interfaces for complex tasks can quickly become complex. We needed to create as natural, invisible and efficient an interface as possible, combining graphical user interface (GUI) techniques when effective with other, more fluid modalities, such as speech and pen.
- The utility of information greatly increases in conjunction with other supporting information. Thus, we required an open and extensible system that places the needed information and tools at the operator's fingertips.

System Description

The first public demonstration of MVIEWS was given at the Exploitation Technology Symposium (ETS-97) held at Naval Research and Development (NRAD) headquarters in San Diego. Describing this event will provide a good overview of how the current MVIEWS implementation can be used.

Given the visuals provided by NRAD's location, we chose to exhibit MVIEWS using a border patrol scenario. After securing the necessary permits, we installed a camera on the roof of the demonstration building, such that it overlooked activity in the harbor below and at a nearby military airport. Our video analyst could investigate the movements of small recreational boats, an occasional commercial or military ship, plenty of windsurfers, and a few airplanes and helicopters.

Inside the exposition's demonstration facilities, an operator was seated in front of a Sun[1] Ultrasparc 1 computer, monitoring the live video feed. The operator interacts with MVIEWS by using pen and voice (Figure 1), to perform one of the following functions:

- *Add annotations*, simply by speaking and drawing directly on top of the video. As an example, a surveillance operator might speak "The movements of

[1] All product or company names mentioned in this document are the trademarks of their respective holders.

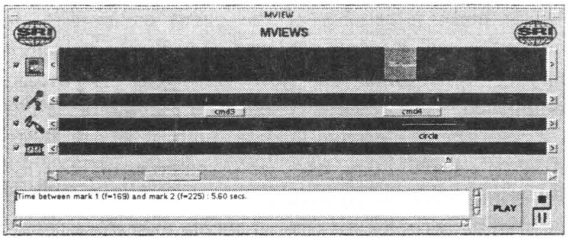


Figure 3. Media Track Editor. A video is replayed with multimedia overlays, and indexed fields are located in the video or the video database.

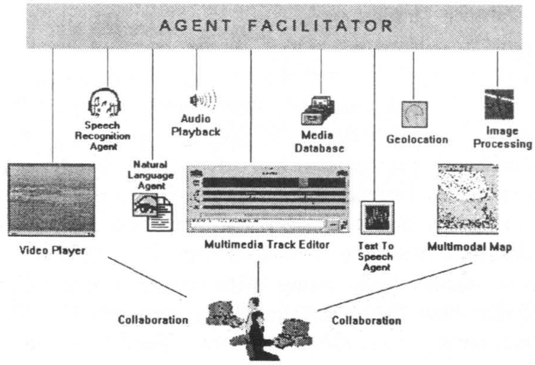


Figure 4. MVIEWS architecture.

this motorboat appear unusual" while drawing a circle around the vessel and tracing its path using the pen.

- *Generate reports*, constructed from multimodal input and multimedia data. A typical interaction might contain: "Report, convoy of three vehicles, heading rapidly west along access road." This information would be saved with the video frame and associated metadata, including time, date, camera type, and spatial coordinates.
- *Specify commands* to the system, involving object tracking (e.g., "Track this boat"), image processing (e.g., "Stabilize image", "Grab this region"), and setting alerts (e.g., "Notify me if this object moves" or "If more than three objects enter this region, alert me").
- *Collaborate* with remote participants, for example "Bob, can you identify this vehicle type?"

In addition to the video display, the operator could call up an interactive map (Figure 2), simultaneously displaying any objects tracked in the video as geo-referenced points in 2D space. The map display, also controlled through pen and voice, provided additional information about the region. For example, in the Predator UAV domain, the map allows the operator to examine the terrain ahead of the plane's path, and call up supplementary data (e.g., "Show me all military bases near here").

As a means of attracting further attention to our ETS demonstration, we sent out a person carrying a wireless handheld pen computer to mingle with a reception gathering nearby. After targeting a small group, the demonstrator would show them the map display, look over the balcony and say, "See that boat down there? It's being automatically tracked by software agents from a live video image, and this computer is receiving the reports. Why don't you go in there and check it out?"

Near the first workstation, we positioned another computer where a second analyst could work collaboratively with the first. While one operator monitored and annotated the continuously advancing live video feed, the second was at liberty to provide more detailed analyses of recorded segments of the video. Using the Media Track Editor (Figure 3) as his primary interface, the analyst navigated within the video segment, requested image enhancements, performed timing and distance measurements, and queried the multimedia database for other video segments of interest. The Media Track Editor is structured as a timeline, with annotations, extracted frames and clips, and recognized speech and pen clearly highlighted. By providing an interface for quickly skimming through a video clip and replaying selected sections along with their multimodal annotations, MVIEWS establishes context for what the analyst is examining, accenting what was important to the operator at the time the video was recorded. Instead of fast forwarding through the entire video clip to understand its content, an analyst can save time by perusing annotations from the timeline at different granularities of detail.

IMPLEMENTATION

To implement MVIEWS, we needed to quickly integrate numerous technologies, written in a variety of programming languages, some requiring specialized computer hardware. To facilitate a loosely coupled, dynamic, heterogeneous and distributed integration, we took advantage of the services provided by the Open Agent Architecture™ (OAA™).[2]

The Open Agent Architecture

Similar in objective to distributed object frameworks such as OMG's CORBA or Microsoft's DCOM, a distributed agent architecture such as the OAA can provide integration of components written in different

[2] For more information about the OAA, see http://www.ai.sri.com/~oaa/

programming languages[3] and running on different platforms.[4] However, OAA agents possess qualities beyond ordinary distributed objects. Agent interactions are more flexible and adaptable than tightly bound IDL[5] method calls in CORBA or DCOM, and are able to take advantage of parallelism and dynamic execution of complex goals. Instead of preprogrammed unitary method calls to known object services, an agent can express its requests in terms of a high-level logical description of what it wants done, along with optional constraints specifying how the task should be performed. This information is processed by one or more Facilitator agents, which plan, execute and monitor the coordination of the subtasks required to accomplish the end goal.

The OAA has been used to implement more than twenty different applications, including

- Multi-robot control and coordination [4]
- Office automation and unified messaging [2]
- Collaborative multimodal user interfaces [1, 12]
- Frontends [8] and backends [11] for the Web
- Development tools [10] for creating and assembling new agents with the OAA

Each OAA project can take advantage of the core services provided by the architecture as well as of the growing number of technologies now accessible through an agent interface. These services and technologies include speech recognition, natural language understanding, text extraction, multimodal fusion and reference resolution, reactive planning, virtual reality, image processing, web-related technologies, user modeling, and collaboration tools.

The core services of the OAA are implemented by an agent library, which has been ported to several different programming languages, working closely with a Facilitator agent, responsible for domain-independent coordination and routing of information and goals. These basic services can be classified into three areas: agent communication and cooperation, distributed data services, and trigger management.

Interagent Communication Language

The Interagent Communication Language (ICL) provides the means for interaction among agents. When an agent wants to make a request of the agent community, it describes the goal it wants achieved as well as parameters specifying constraints on how the goal is to be accomplished. The request is sent to a Facilitator agent, which uses the declarative specifications it stores about each agent's capabilities, and the parameters defined for the incoming goal, to produce a fully specified execution plan detailing tasks for distributed agents to perform. The Facilitator agent is then responsible for monitoring and coordinating the execution of the plan, by routing requests (potentially to agents in parallel), collecting results, backtracking when subgoals fail, and finally providing the results to the requesting agent.

ICL requests are expressed using the syntax and semantics of Prolog, a decision influenced by our desire to involve the user as closely as possible in agent interactions. ICL expressions can be generated from the Prolog-based logical forms produced by many natural language parsers, allowing the user to make requests of the agent community in plain English. As a simple example, the English request *"What is the telephone number of John Bear's manager?"* would be converted to the ICL expression:

```
oaa_Solve( (manager('John Bear', M),
            phone_number(M, P)),
            [query(var(P))] ).
```

Parameters can specify both low-level constraints or high-level advice. Examples of low-level constraints might include the maximum amount of time for the solution, the maximum number of solutions returned for a query, how the information should be returned (e.g., as a blocking call or asynchronous streamed response), and rarely, which agents are allowed to participate in the computation.

As an example of high-level advice parameters, notice that the way parallelism should occur depends on the type of task being solved. If three email agents are available on the network, the request "Send this by email to Luc" should probably not be sent to all three agents at once, but rather if the first doesn't succeed, the others should be tried in succession. Compare this with a database query, where it might be very desirable to send the request in parallel to as many available agents as possible. When solving a math problem, different answers returned by different mathematicians could signal a problem, whereas if the participants are students composing poems, varied answers are a requirement.

Data Management

OAA's distributed data facilities share much in common with the distributed goal resolution process described in the previous section. In the same way that agents register the tasks they are capable of performing, agents also declare descriptions of the data they manage. An agent can then add, delete, change, or query a data value, and

[3] Programming languages: C, C++, Prolog, Lisp, Java, Borland Delphi, and Microsoft Visual Basic.

[4] Platforms: UNIX (SunOs, Solaris, Lynx), Windows (3.1, 95, NT), all Java platforms.

[5] Interface Definition Language: specifies an object's methods using a C++-like header file.

this request will be automatically routed by the Facilitator agent to the appropriate agent or agents.

Data declarations and functions also make use of the notion of parameter lists. In this case, parameters specify information about permissions, scoping, persistence, whether duplicate values are allowed, and so forth. Data parameters are also used provide synchronous collaboration features to OAA applications; the 'shareable' attribute determines whether a data value is synchronized among all participants of a distributed collaborative session.

Triggers

Triggers allow an agent, or set of agents, to monitor some potentially complex state in the world, performing an action if the trigger's test conditions become true.

Triggers or rules exist in many commercial systems today; for instance, mail programs often allow the user to define actions (e.g., delete, archive, forward) to perform if an email of a certain type arrives. However, in these systems, the action must be predefined and fixed. With OAA triggers, the action part of a trigger may be any compound task executable by the dynamic community of agents. As new, perhaps unanticipated, agents connect to the system at runtime, what the user can say and do literally changes. For instance, if a new fax agent suddenly becomes available, the user can now say or write "If email arrives..., fax it to Bill", even if this action had never been conceived of by the original developers of the application.

Four types of triggers are currently defined by the OAA:

1. Data triggers: *"If the airline flight time changes..."*
2. Time triggers: *"In ten minutes...", "every Thursday from now until Christmas..."*
3. Communication triggers: *"If any agent sends Msg..."*
4. Task triggers (specific to the domain of a particular agent): *"If mail arrives about...", "If this Web page changes ..."*

Triggers are stored using the data management facilities, so they can be added, deleted, inspected, protected, and automatically distributed like any other database predicate.

MVIEWS Component Technologies

The MVIEWS application is implemented as a collection of OAA agents, as depicted in Figure 4; we'll now take a closer look at each of the component technologies.

Video Player

Two separate video players have been adapted for use with the MVIEWS system, each with slightly different properties. The first is a public domain UNIX-based program called XAnim, a software-based player capable of displaying numerous video file formats, including MPEG and AVI. The second, MP, was written to take advantage of special libraries provided by Sun to access their hardware video boards. MP can play either from MPEG files or a video source (e.g., live camera or VCR). We are considering integrating yet a third player to provide a Windows PC solution.

The video players were adapted to work within the OAA framework by including the OAA agent library, and by publishing the video players' capabilities using the ICL formalism. In addition, the programs were extended to permit drawing on top of the video by using a mouse or electronic pen.

Speech Recognition

Speech recognition (SR) is used both for entering commands to the system and for extracting content from a user's verbal annotations for video indexing and report generation. The SR technology used by MVIEWS is a large vocabulary, continuous speech, speaker-independent system developed at SRI's STAR Laboratory and then commercialized by a spin-off company, Nuance Communications.[6] The recognizer is based on a hidden Markov model approach, and takes as input a set of models either compiled from a regular expression grammar notation, or constructed through a learning process over a large corpus of data. In the current demonstration system, a grammar defines the set of possible spoken commands, as well as the set of keywords and phrases that can be recognized from annotations or verbal reports.

Natural Language

Two aspects of natural language (NL) processing are used within the MVIEWS system to handle the results from the speech recognition process: NL understanding, to interpret commands to the system, and information extraction from text, to produce indices, summarizations, and reports from spoken annotations.

The OAA is often used to integrate different levels of NL understanding, depending upon the requirements of the system. In most OAA-based systems, prototypes are initially constructed using relatively simple NL components, and as the vocabulary and grammar complexities grow, more powerful technologies can be incrementally added. The current MVIEWS demonstration system has a relatively limited set of commands that are processed by two of our low-end NL systems: Nuance's template-slot tools and DCG-NL, a Prolog-based top-down parser. As the MVIEWS prototype matures, more efficient NL systems can be added, such as

[6] http://www.nuancecom.com/

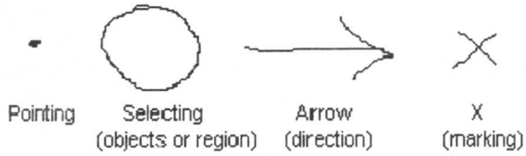


Figure 5. Gesture set.

Gemini, a robust bottom-up parser based on unification grammars, which interleaves syntax and semantics.

A current DARPA-funded project, which will be folded into the MVIEWS system, focuses on the information extraction task in the Predator domain. Applying SRI's FASTUS [6] and adding better domain coverage for speech recognition will be an improvement over the current implementation, which is based on simple keyword spotting and a hand-coded grammar defining possible reported utterances.

Pen Recognition and Annotation

The pen modality is used in conjunction with speech to add multimodal annotations to a video document, and to issue commands to the system. For commands, a set of pen gestures (Figure 5) can be recognized using algorithms developed in [9].

In our experience, most pen annotations made by users also fall into the class of gestures, usually supplemented by a descriptive audio annotation. Since handwriting has rarely been used, incorporating handwriting recognition into the system has been a low-priority task. However, UNIX-based handwriting recognition libraries have been obtained from Communications Intelligence Corporation (CIC[7]), another SRI spin-off company, and may play an important role for labeling objects with out-of-vocabulary names, a task difficult for speech recognition systems.

Image Processing and Object Tracking

The problem with most image processing and object tracking algorithms is that they are often highly specialized, working well for certain situations and not at all in others. An image and video analyst needs to have an entire library of routines at his or her call.

In the current MVIEWS prototype, we have integrated several image functions, such as stabilization and extraction of selected regions, as well as two object tracking algorithms.

The first of the two tracking algorithms is adapted for detecting fast-moving, relatively small objects within specified surveillance regions in the image. This process requires specialized hardware, running on a Datacube Max Video 200 pipeline image processing system (MV200) with a Motorola 68040 host processor. The Lynx operating system on the MV200 is capable of reading and writing directly in the image memories, using a VME bus. The signal from the incoming video stream is digitized into 256 gray levels and then processed at close to 15 frames per second. Each processing step involves detection of motion between adjacent image frames, followed by temporal correspondence to correlate the moving segments in the video sequence. The strength of the approach is in the temporal association process, which is capable of handling occlusions of moving targets and false alarms from the motion algorithm. As moving targets are detected, their position and ID are passed through the OAA to all interested agents.

The second tracking algorithm, running locally on the Sun Ultrasparc workstation, is good at tracking slow-moving objects, given their initial position (e.g., "Track this car", or "If this boat moves, notify me"). This routine works by comparing via convolution a small subimage of the current video frame with a same-sized subimage from the previous frame. Tracking is initiated by selection of a seed subimage covering the object to be tracked. In our experiments, these subimages ranged in size from 11x11 to 27x27 pixels, depending on the size of the object to be tracked. The object model is formed by storing the location of the subimage within the frame along with the pixel values of the subimage and the square root of the sum of all pixel values within the subimage.

To find the location of the tracked object in a new frame, we find the local maximum of convolution scores by shifting the model subimage around the neighborhood of its previous location. When a local maximum is found, the pixel values of the subimage centered at that location in the new frame are taken as the new model. The model is updated every frame. The benefit is that the model can adapt to changing views of the tracked object. The drawback is that the algorithm can sometimes be fooled when a tracked object moves into a region where it cannot be distinguished from the background.

Multimodal Map

The multimodal map (MMAP) component of the MVIEWS system allows a user to interact with a dynamic map display through a natural combination of speaking, writing, and drawing directly on the map surface. Multiple modalities may be entered simultaneously or in any sequential order, and merged to produce a command or request. This fusion makes use of the inherent parallelism of the OAA, with multiple agents competing and cooperating to resolve ambiguities arising during the interpretation process.

[7] http://www.cic.com/

The multimodal map has been used in various OAA applications, such as providing a natural user interface to travel-related sources on the Web [1], and for guiding and monitoring multiple mobile robots [4]. Adding MMap's functionality as part of the MVIEWS application demonstrates OAA's extensibility and flexibility, in that no code had to be written to incorporate the technology into the MVIEWS domain.

Human Collaborative Tools

Within the OAA, a human user will typically interact with distributed software agents, and the agents themselves will communicate and cooperate with each other. A natural extension to this paradigm is to allow multiple human users to work with each other in a collaborative fashion.

The OAA has been extended to include services that facilitate adding synchronous collaborative functionality to any OAA-based user interface. The session management, state replication and an activity-based floor mechanism are provided in a peer-to-peer topology by the Synchronous Collaborative Object Oriented Toolkit (SCOOT) [3], developed by SRI's Augmented Collaboration Group, or alternatively by a purely OAA-based collaboration mechanism (centralized).

Lessons Learned

Although we have not yet run formal experiments to evaluate the utility of the MVIEWS system, our experiences do suggest several lessons.

The first is simply that that there is a strong need in the Intelligence community for better tools to help an analyst interact more efficiently with video. Video's role in the analysis process is growing: in the case of the Predator UAV, its camera was originally intended strictly as sensor for the pilot to guide the plane, but ended up playing a role in battlefield assessment. In general, the MVIEWS concept has been well received at ETS and elsewhere, and many viewers have suggested domains to which it could be applied.

On the implementation side, two important questions were: how much and what kind of information should flow among distributed agents; and how should we deal with inaccuracies in technologies such as speech recognition or image processing.

Regarding information routing, we chose to limit interagent exchange to messages containing semantic representations of the data, not the data itself (instead of moving video across a network, we split the physical cables so each machine could have a local input source). Agents registered triggers, filters and constraints on broadcast data (as described in the section on OAA) and simple heuristics were inserted for regulating the rate with which information needed to be updated: a fast moving object requires more frequent updates, than a slow one. Clearly, our simple prototype has not solved all hard problems in this area; however the facilitated approach seems promising for managing an efficient data flow.

Regarding recognition technologies, we found that speech and pen are currently reliable enough to be of use for controlling tasks in the user interface, but that for other tasks (speech: transcription of annotations; image processing: object tracking), recognition technologies produce imperfect results given a broad class of input. We chose to design MVIEWS such that as these technologies mature, they can take on an increased role. Multimedia annotation and playback are reliable and useful features by themselves, and provide data on which transcription and classification technolgies can act in the future. An analyst can apply image functions such as stabilization and enhancement with confidence, and then get a sense of when this can be augmented by object tracking or detection. With image processing, a single algorithm will not be sufficient for all input, so we incorporated multiple algorithms, with their use under the direction of the human user. Eventually, we may be able to automate the decision about under which conditions each should be used.

RELATED WORK

Commercially available tools provide much useful functionality for video manipulation and annotation. One good example is Z/Videoware from Z Microsystems.[8] Z/Videoware allows a user to play digital video clips and add audio annotations. It possesses an innovative feature that tries to classify the video by examining the closed-caption text embedded within, and then sorting the video appropriately into different folders.

Another example of a system for video annotation and analysis is VANNA [5]. This application provides a highly tailorable user interface, and an efficient system for annotating the video, using a variety of input devices, such as mouse, trackball, keyboard, touch screen and electronic pen.

Although commercial systems provide some of the functionality that one would want for video annotation, they are missing features that we feel are required for effective video exploitation, such as collaboration, natural user interfaces, and the ability to call up a variety of related data and tools from the same user interface. Although several research systems attempt to apply more advanced technologies to image processing [13], we are not aware of such systems focusing on video.

One effort that has a great deal in common with MVIEWS

---

[8] Z Microsystems' homepage: http://www.zmicro.com/

is a DARPA-funded project called QuickTurn [7], by MITRE and Carnegie Mellon University. MVIEWS and QuickTurn share many of same goals (an integrated environment combining advanced user interfaces with tools and databases for the intelligence analyst) and technologies (collaboration, mediated databases, maps, image tools). However, MVIEWS attempts to focus its solutions within the context of the video analyst (e.g., object tracking, indexed search on pen and voice annotations).

CONCLUSIONS AND FUTURE DIRECTIONS

The current version of MVIEWS can only be considered a working prototype system, but this IR&D project has already shown potential for enhancing the tools and techniques available in the domain of video exploitation and analysis. By using an open, distributed approach as the underlying architecture, we were able to rapidly bring together pertinent technologies, and facilitate the future development of the system as components are added, improved or replaced. Although MVIEWS consists of a number of capabilities that are interesting by themselves, it is the *integrated use* of these capabilities that can greatly enhance the effectiveness of the operator.

Improvements and extensions will be made to the system. One comparatively large project is already under way to improve the speech recognition and data extraction from annotations in the Predator domain, by combining robust speech recognition and the FASTUS text extraction software. Other improvements to MVIEWS will involve a wider use of the collaboration technology, adding more object tracking and image processing techniques, and identifying and integrating additional video-related technologies. We will also pursue the opportunity to perform user experiments to better quantify benefits of the system.

ACKNOWLEDGMENTS

This IR&D-funded project is the result of contributions from many talented people spanning six centers within SRI. The participants include: Jeff DeCurtins, Gopalan Ravichandran, Bikash Sabata (video and image processing), Greg Myers, Bob Bolles, Eric Rickard, Joel Cain (vision, design and direction), Luc Julia, Adam Cheyer (agent architecture, speech, gesture, handwriting).

REFERENCES

1. Cheyer, A. and Julia, L. Multimodal maps: An agent-based approach. In *Proc. of the International Conference on Cooperative Multimodal Communication (CMC/95)*, Eindhoven, May 1995.

2. Cohen, P., Cheyer, A., Wang, M., and Baeg, S. An open agent architecture. In *AAAI Spring Symposium*, pages 1—8. Stanford University, March 1994.

3. Craighill, E., Fong, M., Skinner, K., Lang, R., and Gruenefeldt, K. SCOOT: An Object-Oriented Toolkit for Multimedia Collaboration. In *Proc. ACM Multimedia '94*, pages 41—49, San Francisco, October 1994.

4. Guzzoni, D., Cheyer, A., Julia, L., and Konolige, K. Many Robots Make Short Work. *AI Magazine*, Vol. 18, Number 1, pages 55—64. Spring 1997.

5. Harrison, B. L. and Baecker, R. M. Designing Video Annotation and Analysis Systems. In *Proc. of the Graphics Interface 92 Conference*, pages 157—166. Vancouver, B.C., May 11-15, 1992.

6. Hobbs, J., Appelt, D., Bear, J., Israel, D., Kameyama, M., Stickel, M., and Tyson, M. "FASTUS: a cascaded finite-state transducer for extracting information from natural-language text," in *Finite State Devices for Natural Language Processing* (E. Roche and Y. Schabes, eds.), Cambridge MA: MIT Press, 1996.

7. Holland, Roderick. QuickTurn: Advanced Interfaces for the Imagery Analyst. *DARPA/ITO Intelligent Collaboration & Visualization (IC&V) Program PI Meeting*. Dallas, Texas, October 10, 1996. http://www.ito.darpa.mil/Proceedings/icv/agenda.html

8. Julia, L., Cheyer, A., Neumeyer, L., Dowding, J., and Charafeddine, M. HTTP://WWW.SPEECH.SRI.COM/DEMOS/ATIS. In *AAAI Spring Symposium*, pages 72—76. Stanford University, March 1997.

9. Julia, L., and Faure, C. Pattern recognition and beautification for a pen-based interface. In *ICDAR'95*, pages 58—63, Montreal, Canada, 1995.

10. Martin, D., Cheyer, A., and Lee, GL. Agent development tools for the open agent architecture. In *Proc. of the International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM)*. London, April 1996.

11. Martin, D., Oohama, H., Moran, D., and Cheyer, A. Information brokering in an agent architecture. In *PAAM'97*. London, April 1997.

12. Moran, D., Cheyer, A., Julia, L., and Park, S. Multimodal user interfaces in the Open Agent Architecture. In *Proc. of IUI-97*, pages 61—68. Orlando, Jan. 1997.

13. Srihara, R., Zhang, Z., and Chopra, R. Show & Tell: Using Speech Input for Image Interpretation and Annotation. In *AAAI-97 Spring Symposium, Workshop on Intelligent Integration and Use of Text, Image, Video and Audio Corpora*, pages 17—24. Stanford University, March 1997.

ATTORNEY OR PARTY WITHOUT ATTORNEY (Name, State Bar number, and address)
J. DAVID HADDEN
FENWICK & WEST LLP
801 CALIFORNIA STREET MOUNTAIN VIEW, CA 94041

TELEPHONE NO.: (650) 988-8500 | FAX NO. (650) 938-5200 | E-MAIL ADDRESS (Optional):
ATTORNEY FOR (Name): :

FOR COURT USE ONLY

UNITED STATES DISTRICT COURT FOR THE DISTRICT OF DELAWARE

STREET ADDRESS:
MAILING ADDRESS:
CITY AND ZIP CODE: , DE
BRANCH NAME: DELAWARE

Hearing Date: 8/1/2019 Room:
Hearing Time: 9:00 AM Dept:

PLAINTIFF: IPA TECHNOLOGIES, INC.

DEFENDANT: AMAZON.COM, INC., AND AMAZON DIGITAL SERVICES LLC

CASE NUMBER:
1:16-CV-01266-RGA

Ref. No. or File No.:

PROOF OF SERVICE

AT THE TIME OF SERVICE I WAS AT LEAST 18 YEARS OF AGE AND NOT A PARTY TO THIS ACTION
I SERVED COPIES OF THE FOLLOWING DOCUMENTS:

1. COVER LETTER; 2. NOTICE OF SUBPOENA TO SRI INTERNATIONAL; 3. SUBPOENA TO PRODUCE DOCUMENTS, INFORMATION. OR OBJECTS OR TO PERMIT INSPECTION OF PREMISES IN A CIVIL ACTION; 4. EXHIBITS A-K

PARTY SERVED: **SRI INTERNATIONAL**

PERSON SERVED: **DAVID STRINGER-CALVERT - AGENT**

DATE & TIME OF DELIVERY: **7/18/2019**
12:10 PM

ADDRESS, CITY, AND STATE: **333 Ravenswood Ave**
Menlo Park, CA 940253453

PHYSICAL DESCRIPTION: **Age: 30** **Weight: 180** **Hair: BLONDE/BALD**
Sex: Male **Height: 6'0**
Skin: CAUCASIAN

MANNER OF SERVICE:
Personal Service - By personally delivering copies.

WITNESS FEES:
Were offered or demanded and paid: $46.00.

Fee for Service: $ 370.00
Registration No.:
County:
JPL PROCESS SERVICE, LLC
14482 BEACH BLVD. STE S
WESTMINSTER, CA 92683
(866) 754-0520

I declare under penalty of perjury under the laws of the The State of California that the foregoing information contained in the return of service and statement of service fees is true and correct and that this declaration was executed on July 18, 2019.

Signature: ______________
BENJAMIN STURGES

PROOF OF SERVICE

Order#: 141659/General

Exhibit 12

Moran CV extended
2003

Exhibit 1

EXHIBIT 2020
Deponent Moran
Date 11/21/19
Rebecca Romano, CSR #12546

Douglas B. Moran

790 Matadero Avenue
Palo Alto, CA 94306-2734
Home: +1-650-856-3302
E-mail: dm .nospam @dougmoran.com

Table of Contents

Employment History

September 2001 - present: Independent consultant. Various projects, evaluations, reports, proposals.
May 1999 - August 2001: Recourse Technologies, Inc: a start-up developing computer security software for Threat Management. Acquired by Symantec in August 2002. Products: *ManTrap* (a honeypot), *ManHunt* (Intrusion Detection and DDoS response), and *TipOff* (forensics, not released).
Title: Vice President, Research and Development

- Early employee (#4), with consequent contributions to early growth of company.
- System architect and lead programmer for the TipOff product. TipOff was based on the approach that I developed under the DERBI project at SRI. Seven patent applications filed.
- Supported the development of the other products, primarily in requirements, testing and evaluation. Used contacts to acquire sophisticated test data sets that found bugs not exercised by the existing testing.
- Supported the marketing activities for the company and the individual products. Wrote the basic whitepapers for both ManTrap and ManHunt and both of the company's published papers. Assisted Marketing Communications by identifying themes that would appeal to journalists and by translating technical details into appropriate descriptions and analogies. Significant contributor to the company's basic positioning.
- Supported sales by aggressively evangelizing the products to my contacts, many of whom were in the position of "evaluators" or "influencer." Significant sales resulted.
- Created and championed a plan to greatly expand the market for the ManTrap product by making it practical and useful to deploy more and larger clusters of them. Was pursuing government funding for much of this work.

1983-1999: SRI International (formerly Stanford Research Institute)
Note: At SRI, researchers routinely work on multiple projects, with senior staff having leadership/management roles on their primary projects.
Title: Senior Computer Scientist

Departments:

- 1987-1999: Artificial Intelligence Center: widely regarded as a world class research group in AI

1986-1987: Cambridge (England) Computer Science Research Centre: Co-founder of SRI's first non-US research facility (others: Dr. Robert C. Moore, now of Microsoft Research and Dr. Hiyan Alshawi, now of AT&T Labs-Research).
- 1983-1986: Telecommunications Sciences Laboratory

Projects

- 1996-1999: Diagnosis, Explanation and Recovery from Break-Ins (DERBI): A host-based intrusion detection system (HIDS) that assumed an out-of-the-box configuration (no special tools or settings). In DARPA IDS Evaluations (1998 and 1999), it used only end-of-day dumps as its data, but its performance was comparable to HIDS that used full real-time auditing data (Sun's BSM). Although it was also applicable to realtime and near-realtime detection, DERBI focused on after-the-fact detection. To overcome the intruder's attempts to camouflage his presence and actions, DERBI used information from a wide range of sources to identify likely times and elements of the intrusion, and then use that information to identify additional elements that could be part of the intrusion.
    - I created the project and obtained funding (DARPA), and then was system architect, lead programmer and project leader. Team of 3-6.
    - Final Report: *http://www.dougmoran.com/dmoran/PAPERS/DerbiFinal.pdf*

- 1994-1998: Open Agent Architecture™ (OAA). A distributed multi-agent system that focused on rapid assembly of new systems from existing applications. Included a multimodal user interface (speech, natural language, gesture and WIMPS). It was inspired by the experiences of the ShopTalk project (below).

  The focus was facilitating the creation of systems from components that were not intended to work together by providing a significantly higher level of abstraction than that found in objected-oriented methodologies and Object Request Brokers. An SDK simplified "wrapping" existing (legacy) applications to transform them into OAA agents. The UI was implemented with agents for each supported modality, with multiple alternatives for most modalities. Because the UI was implemented with agents, the user appeared to be just another agent to the rest of the application, eliminating the need for the non-UI agents to differentiate between interactions with the user *vs.* automated agents.

    - Co-founder, subsequently Project Leader/Supervisor for suite of related projects. Team of 3-7 SRI staff, plus 1-4 visitors from sponsors (primarily Japanese and Korean).
    - Focused on management, marketing, sales, and customer relations, winning contracts from a range of clients (government and commercial, US and international).
    - Formulated technical requirements and approaches, but being the only senior staff member, I had to forgo participating in the implementation.
    - Honed the marketing message (presentation and demonstration) and technology to the extent that the OAA became one of SRI's *"Top 10"* technologies. My ability to quickly customize the presentation to a particular client or audience resulted in it being heavily used to promote SRI in general, including a presentation to a head-of-state.
    - The success of the OAA design and implementation was testified to by its incorporation into a range of other SRI systems.
    - Featured in "*Computing Goes Everywhere*", *Technology Review* magazine, January 2001.
    - Videos at *http://www.ai.sri.com/~oaa/distrib.html#videos*.

  This project fell victim to its own successes. The team was repeatedly recognized for doing an outstanding job creating demonstration systems highlighting the potential of various SRI technologies, and this became the primary focus of the project. Although there was broad agreement that further research needed to be done on issues related to large-scale systems, staffing decision made those issues difficult to pursue, and I chose to leave the project and focus on the DERBI system.

- 1985-1992: ShopTalk: An evolving series of research prototypes that demonstrated the effectiveness of multimodal user interfaces. The user interface provide transparent, integrated access to multiple applications, for example, processing a query could involve combining historical data from a database with projected results from

a simulator. This project arose from the merger of two related activities: a colleague's established marketing activity and my robust implementation of a prototype containing many of those ideas (as part of the CCWS project (below)).
    - Deputy project leader, focusing on internal activities: lead programmer and researcher. Team of 4-8.
    - Substantial contribution to marketing activities: Created and customized presentations and demonstrations, presenting part or all to many audiences.

- 1987-1994: Spoken Language Systems: The *Gemini* system was based on the CLE system (below), re-implementing many of the algorithms thereby allowing application of "lessons learned" and design changes to better accommodate integration with existing speech recognizers.
    - Researcher and developer focusing on the integration of speech recognition and natural language understanding technologies. The speech recognizer generated a large weighted set of possible word sequences, but the initial Gemini system expected a single input and produced an ordered (but unweighted) list of results.
- 1986-1987: Core Language Engine (CLE): A highly capable natural language processing system that integrated promising new technologies and approaches while applying the lessons learned from earlier large-scale systems.
    - Researcher and developer, contributing to the development of the overall system, with primary responsibility for the pragmatics component (quantifier scoping).
- 1986-1988: Delphi: course-grained OR-parallelism in logic programs.
- 1983-1986: Command & Control Workstation (CCWS): Design and implement multimedia mail and multimodal conferencing applications. System was installed on an aircraft carrier (*USS Carl Vinson*) and used during an operational deployment.
    - Project leader (replacing original) for a team of 6-7.
    - Designer and implementer of automated agent in the conferencing system that added graphical information to the conference workspace using spoken language access to databases.
    - Developed a commercial quality GUI toolkit. A software vendor (Quintus Computer Systems, acquired by Avaya 4/2001) explored licensing it for sale, and although it was judged to be technically superior (functionality, maturity, stability), they chose to license one of the alternatives for unspecified "business reasons."
- Computer Facility (1983-1994) (part-time): This activity provided the practical experience that led to the DERBI project and the TipOff product (above).
    - Developed extensive collection of tools to simplify the administration of a rapidly growing computer facility. The initial set of processes and tools was designed for my project's Sun workstations, but they were so successful that other projects arranged to have their computer become part of my cluster, resulting in my becoming manager for all the department's Suns. The Sun cluster scaled from an initial 4 workstations to 60 without an increase in support staff.

1980 - 1983: Oregon State University, Assistant Professor in Dept. of Computer Science.

- Teaching effectiveness: During those 3 years, my primary undergraduate programming course grew from an initial enrollment of 50 to 400, with a substantial wait-list.
- Computer facility: As part of move of the department, supervised construction of computer rooms and upgrade and expansion of the computer equipment.

1974 - 1979: The University of Michigan (Ann Arbor), Department of Computer and Communication Sciences

- Teaching Assistant: Head TA for the second course on programming. Taught recitation sections for that course and others.
- Research Assistant: Transformed an abstract system based on formal logic into a computationally viable system, and in the process unified what had been distinct components.

    Details: I was responsible for the semantic component of a project that produced a computationally viable version of a sophisticated formal linguistic theory. The semantics were based on formal mathematical models, and the smallest non-trivial model for the basic theory had (2^2^523) elements, but only tens of those elements were

actually used. I developed a system that allowed the models to be incrementally built and used without introducing inconsistencies. This system was subsequently used a several universities as both a teaching and research tool.

As part of this work, I recognized that the linguistic phenomena that had been described as a separate topic could be unified with semantics as a side-effect of a *process-oriented* treatment: the sentence being processed could add information to the knowledge base (the model) and consistency checks on the new information needed to be performed "on the fly."

- President of the Graduate Employees Organization (1976). The GEO represented the 2100 graduate students working as teaching, research and staff assistants. Learned meeting management skills and honed advocacy writing.

Computer Security Activities

My introduction to computer security came as a side-effect of my managing a computer facility. Through the early 1990's, SRI was a high-profile site on the Internet and hence was a frequent target for hackers. Diagnosing and tracking back attacks was labor-intensive, and hence slow. Too often the trail would go cold because of the substantial delays in tracking an attack through even a small number of sites. The DERBI research project resulted from the realization that a large portion of this process could be automated, making diagnosis and trackback more effective (faster and more accurate). I then joined Recourse Technologies to commercialize this approach.

Representative sample of activities in this area:

1. Manual trackback of attacks (most of my activity)
   - Wrote an extensive collection of tools to simplify the administration of the computer cluster. These tools provided notification of inconsistencies, developing problems and other anomalous situations, and, as a side-effect, revealed a variety of probes and attacks (both attempted and successful). My successor stated that the most important lesson I taught him was *"Good system administration is good security, and vice versa."*
   - Aggressively pursued evidence of hacker activity. We collected evidence of attacks, analyzed it and reported the results. We contacted sites that the attacker was using to launch attacks and passed on our knowledge of the attacker's tools and techniques.

     We decided to not ignore probes and unsuccessful attacks because our group had collaborative relationship with organizations with poor security practices: By taking countermeasures against attackers we would somewhat decrease the probability of those other sites being compromised, which would in turn decrease the risk of one of those sites being used to compromise ours.

   - Established a reputation for quickly detecting hacker activity that had gone unnoticed in other parts of the company. Became lead for company-wide computer security activities.
   - Most notable instance: In August 1993, we detected an early instance of Rootkit. We captured much of the toolkit, including the sniffer. We tracked back the attacks to other compromised sites and worked with them to extend the trackback. We were extremely alarmed to find attempts against hosts that were two of the root name servers. We notified CERT and worked with the FBI. The FBI told us they thought they had identified the culprit, and asked us not to go public while they investigated. I wrote and distributed a sniffer-detector to affected sites and to CERT. In early November, a story appeared in the New York Times (by John Markoff) about thousands of computers that had been compromised by this attack. Aside: CERT did not release an advisory (CA-1994-01) until February 3, and it contained a sniffer detector similar to the one I had provided to them.
     We received no public credit, although from everything I could determine, we were the first to report this attacker.
2. Interaction with CERTs and law enforcement agencies

- Reports and follow-up of attacks (above)
- Comments on requirements and effectiveness
  - Critique of that became part of documentation for the need for CERT (below)
  - Critiques on non-responsiveness of CERT, for example, "security incident" handling -- comments on CERT's policy from Risks Digest, 15, 18 (27 October 1993) (*http://catless.ncl.ac.uk/Risks/15.18.html#subj6.1*)

3. Expert consultant and witness
   - 1999: Consultant for attorney for corporation (defendant) in suit claiming unwarranted disciplinary action based upon allegation of computer misuse.
   - 1996: Consultant and witness for San Mateo District Attorney. A system administrator fabricated evidence of misconduct by his supervisor, causing him to be fired. Featured in a page 1 story in the October 11, 1997 Washington Post: "*Mainframed: Computers Become a Weapon in Office Warfare*" (by John Boudreau).
   - 1996: Consultant for San Mateo District Attorney. Provided independent assessment of data being provided by a corporation's system administrators as evidence in a felony case.
   - 1993: Consultant for Santa Clara District Attorney. Defendant pleaded guilty.
   - Consultant for attorney for a student charged by his university with hacking university computers and a corporate network. Charges were dismissed.
4. Reports and other feedback on security problems to vendors.
   - Example: REXD (1987): First report of vulnerability to Sun. Provided a demonstration of its severity that motivated Sun to deactivate this server in subsequent releases.
   - Morris Worm follow-up (1988): Authored summary of pattern of known vulnerabilities being allowed to persist from release to release. Sent as letter by SRI CEO to Sun CEO (McNealy). Was also used by DARPA (Bill Sherlis) as part of the justification for the creation of CERT.
5. Public distribution of patches and scripts for various vulnerabilities and configuration problems. Except for the simplest cases, we worked through an intermediary, such as the vendor or CERT, because we lacked the resources needed for answering questions and the other inevitable follow-up. Most visible examples of these distributions:
   - TFTP CHROOT patch for SunOS (1986): subsequent SunOS release incorporated the CHROOT capability.
   - Anonymous FTP installation script automating extended sequence of steps described in manual. Modified version subsequently distributed by CERT.

Areas of Technical Expertise

- Computer Security, Multi-agent Systems, Artificial Intelligence, Natural Language Processing
- *Primary programming languages:* C, Prolog
- *Primary scripting languages:* PERL and various UNIX languages (shells, awk, sed, ...)
- *Operating Systems:*
  - UNIX & variants:
    - System programmer (non-kernel), system administrator, expert user
    - UNIX: from V7 to BSD 4.x to SunOS 0.7 to SunOS 5.8 (Solaris 8)
    - Linux: from RedHat 5.2
    - Limited usage of other variants
  - Microsoft Windows: advanced user (W95, W98, NT, W2000 Professional)

Education

University of Massachusetts (Amherst),

Fellow, A.P. Sloan Foundation grant for Interdisciplinary Studies in Cognitive Sciences in Dept. of Linguistics and Dept. of Computer and Information Sciences.

The University of Michigan (Ann Arbor),

M.S. and Ph.D. from the Dept. of Computer and Communication Sciences.
Concentration: Artificial Intelligence and Computational Linguistics

Dissertation: *Model-Theoretic Pragmatics: Dynamic Models and an Application to Presupposition and Implicature*

Massachusetts Institute of Technology,
Bachelor of Science from the Computer Science program in the Dept. of Electrical Engineering (VI-3).

Professional Societies

American Association for Advancement of Science (AAAS)

American Association for Artificial Intelligence (AAAI)

Association for Computing Machinery (ACM)

IEEE and IEEE Computer Society

Sigma Xi, The Scientific Research Society

Other Activities

Neighborhood Association: Barron Park, Palo Alto (approx 1600 households): President, e-mail list manager and co-webmaster (*http://www.bpaonline.org/*)

Other formats and additional information: see *http://www.dougmoran.com/dmoran/*.
This document is available in multiple formats (HTML, PDF, RTF, MS-Word and plain text).
A short version of this resume is available, as well as some additional detail.

Version: *$Date: 2002/09/26 21:39:03 $*

SRI webpage 1998

Douglas B. Moran


```
Douglas B. Moran              Title:      Senior Computer Scientist
AI Center, EJ233              E-mail:     moran@ai.sri.com
SRI International             Telephone:  +1-650-859-6486
333 Ravenswood Ave.           FAX:        +1-650-859-3735
Menlo Park, CA 94025-3493
```

Research Areas

- Intelligent Agents / Multiagent Systems
- Multimedia/Multimodal Interfaces
- Intrusion Detection / Computer Security
- Natural Language Understanding
- Integrating Speech Recognition and Natural Language
- (Prolog and Logic Programming)

Projects

- Open Agent Architecture (includes Multimedia/Multimodal Interfaces) papers
- Diagnosing, Explaining and Recovering from Break-Ins (DERBI)

Earlier Projects

- Spoken Language Systems (1986-1994)
- Shoptalk (1985-1992): papers, presentation transparencies
- Delphi (OR-parallelism in logic programs) (1987-1988): papers
- Command & Control Workstation (multimedia) (1983-1986): papers
- Multimedia Conferencing (1985-1986, part of CCWS)
- Montague Grammar (formal semantics for English) (1975-1981): papers
- UNIX Computer Facility Administration/Management (1982-1993)

Publications

- Formatted bibliography
- Publications list in raw (bibtex) form
- Relevant Bibliography Keywords: for use in searching the bibliographic database (above)

Previous Locations

Other Activities

Mirrored Papers (by others): Belief, Desire, Intention

Directions for Visitors to the AIC

Getting Around: AIC Home Page, AIC Staff

---

moran@ai.sri.com
3 April 1997

Publications

Publications: Douglas B. Moran

Computer Security

D. B. Moran, "*Effective Deployment of Honeypots against Internal and External Threats*," Information Security Bulletin, October 2000.

D. B. Moran, "*Trapping and Tracking Hackers: Collective security for survival in the Internet Age*," Proceeding of the Third Information Survivability Workshop, Boston, Massachusetts, October 24-26, 2000. *http://www.dougmoran.com/dmoran/PAPERS/isw2000.htm* and *http://www.dougmoran.com/dmoran/PAPERS/isw2000.pdf*
Presentation transparencies: *http://www.dougmoran.com/dmoran/PAPERS/isw2000-slides.pdf*

W. M. Tyson, "*DERBI: Diagnosis, Explanation and Recovery from Computer Break-ins*", Final Report, Artificial Intelligence Center, SRI International, January 12, 2001. DARPA Project F30602-96-C-0295. *http://www.dougmoran.com/dmoran/PAPERS/DerbiFinal.pdf*

U. Lindqvist, D. B. Moran, P. A. Porras, W. M. Tyson, "*Designing IDLE: The Intrusion Data Library Enterprise*," First International Workshop on the Recent Advances in Intrusion Detection (RAID), September 14-16, 1998, Louvain-la-Neuve, Belgium. (Short presentation, full paper presented at NORDSEC'98 (below)). *http://www.raid-symposium.org/raid98/Prog_RAID98/Talks.html#Lindqvist_15* and *http://www.dougmoran.com/dmoran/PAPERS/RAID98-IDLE.ps*

Douglas B. Moran, "*Intrusion Detection in the Large: Distributed Detection of Distributed Attacks,*" Presentation to Computer Misuse & Anomaly Detection, Monterey, California, 12-14 November 1996. (PDF of transparencies). *http://seclab.cs.ucdavis.edu/cmad/4-1996/pdfs/Moran.pdf*

U. Lindqvist, D. B. Moran, P. A. Porras, W. M. Tyson, "*Designing IDLE: The Intrusion Data Library Enterprise,*" NORDSEC'98 (Third Nordic Workshop on Secure IT Systems), November 5-6, 1998, Trondheim, Norway. PostScript and PDF

Open Agent Architecture

D. L. Martin, A. J. Cheyer, and D. B. Moran, "*The Open Agent Architecture: A framework for building distributed software systems*," *Applied Artificial Intelligence*, vol. 13, pp. 91-128, January-March 1999. *http://www.dougmoran.com/dmoran/PAPERS/oaa-aai1999.pdf*

D. L. Martin, A. J. Cheyer, and D. B. Moran, "*Building distributed software systems with the Open Agent Architecture*," in *Proc. of the Third International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology*, (Blackpool, Lancashire, UK), The Practical Application Company Ltd., March 1998. *http://www.dougmoran.com/dmoran/PAPERS/oaa-paam1998.pdf*

D. L. Martin, H. Oohama, D. Moran, and A. Cheyer, "*Information brokering in an agent architecture*," in *Proceedings of the Second International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology*, (London), The Practical Application Company Ltd., April 1997. *http://www.dougmoran.com/dmoran/PAPERS/oaa-paam1997.pdf*

Abstract: To date, document identification based on keyword matching strategies has been the basis of most efforts to provide assistance in accessing information resources on the Internet. However, in view of

the limitations on human browsing time and the evolution of more capable software agents, we can expect rapid expansion in the use of fully queryable information sources (such as databases and knowledge bases) and semiqueryable sources (such as form-based query pages on the World Wide Web, and collections of Web pages that are structured by textual markups). ... This paper describes a working prototype Information Broker system, developed within the Open Agent Architecture framework, that provides transparent access to a variety of information sources, each encapsulated as an independent agent. ...

D. B. Moran, A. J. Cheyer, L. E. Julia, D. L. Martin, and S. Park, "*Multimodal user interfaces in the Open Agent Architecture*," in *Proc. of the 1997 International Conference on Intelligent User Interfaces (IUI97)*, (Orlando, Florida), pp. 61-68, 6-9 January 1997. Copyright 1997 ACM (see http://www.acm.org/pubs/copyright_policy/). *http://www.dougmoran.com/dmoran/PAPERS/oaa-iui1997.pdf*

D. B. Moran and A. J. Cheyer, "*Intelligent agent-based user interfaces*," in *Proceedings of International Workshop on Human Interface Technology 95 (IWHIT'95)*, (Aizu-Wakamatsu, Fukushima, Japan), pp. 7-10, The University of Aizu, 12-13 October 1995. *http://www.dougmoran.com/dmoran/PAPERS/oaa-iwhit1995.pdf*

Abstract: We have developed, and are extending, a multimodal human-computer interface in which the selection of modalities to be used takes into account the available resources and the user's current environment and preferences. Our current work focuses primarily on interpreting multimodal inputs and adapting to available output modalities. We anticipate providing more intelligent production of multimodal output in future systems. We have an agent-based system architecture that we use to allow us to incrementally add functionality, and to substitute alternative handling of individual modalities.

Spoken Language Systems

R. Moore, D. Appelt, J. Dowding, J. M. Gawron, and D. Moran, "*Combining linguistic and statistical knowledge sources in natural-language processing for ATIS*," in *Proceedings ARPA Spoken Language Systems Technology Workshop*, (Austin, Texas), 22-25 January 1995. *http://www.dougmoran.com/dmoran/PAPERS/gemini-arpa1995.pdf*

Abstract: During the past year, significant improvements have been made in the natural-language processing technology used in the SRI ATIS spoken-language understanding system. The principal developments have been (1) the incorporation of information from the natural-language grammar and lexicon into a statistical language model that is used in both speech recognition and language understanding, (2) implementation of a robust interpretation component that constructs queries out of grammatical fragments when an utterance cannot be analyzed as a single phrase or utterance, and (3) a new context mechanism for air travel planning that constructs an explicit model of the user's intended itinerary.

J. Dowding, R. Moore, F. Andry, and D. Moran, "*Interleaving syntax and semantics in an efficient bottom-up parser*," in *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, (New Mexico State University, Las Cruces, New Mexico), pp. 110-116, June 1994. *http://www.dougmoran.com/dmoran/PAPERS/gemini-acl1994.pdf*

Abstract: We describe an efficient bottom-up parser that interleaves syntactic and semantic structure building. Two techniques are presented for reducing search by reducing local ambiguity: Limited left-context constraints are used to reduce local syntactic ambiguity,and deferred sortal-constraint application is used to reduce local semantic ambiguity. We experimentally evaluate these techniques, and show dramatic reductions in both number of chart edges and total parsing time. The robust processing capabilities of the parser are demonstrated in its use in improving the accuracy of a speech recognizer.

R. Moore, M. Cohen, V. Abrash, D. Appelt, H. Bratt, J. Butzberger, L. Cherny, J. Dowding, H. Franco, J. M. Gawron, and D. Moran, "SRI's recent progress on the ATIS task," in *Proceedings ARPA Spoken Language Systems Technology Workshop*, (Merrill Lynch Conference Center, Princeton, New Jersey), 6-8 March 1994.

J. Dowding, J. M. Gawron, D. Appelt, J. Bear, L. Cherny, R. Moore, and D. Moran, "*GEMINI: A natural language*

system for spoken-language understanding," in *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, (Ohio State University, Columbus, Ohio), pp. 54-61, 22-26 June 1993. http://www.dougmoran.com/dmoran/PAPERS/gemini-acl1993.pdf

J. Dowding, J. M. Gawron, D. Appelt, J. Bear, L. Cherny, R. Moore, and D. Moran, "Gemini: A natural language system for spoken-language understanding," Technical Note 527, AI Center, SRI International, 333 Ravenswood Ave., Menlo Park, CA 94025, April 1993.

Abstract: Gemini is a natural language understanding system developed for spoken language applications. The paper describes the architecture of Gemini, paying particular attention to resolving the tension between robustness and overgeneration. Gemini features a broad-coverage unification-based grammar of English, fully interleaved syntactic and semantic processing in an all-paths, bottom-up parser, and an utterance-level parser to find interpretations of sentences that might not be analyzable as complete sentences. Gemini also includes novel components for recognizing and correcting grammatical disfluencies, and for doing parse preferences. This paper presents a component-by-component view of Gemini, providing detailed relevant measurements of size, efficiency, and performance.

H. Murveit, D. Appelt, C. Barker, J. Bear, J. Butzberger, J. Dowding, E. Jackson, D. Magerman, R. Moore, D. Moran, A. Podlozny, P. Price, and M. Weintraub, "*SRI's speech and natural language evaluation*," in *Proceedings Fourth DARPA Workshop on Speech and Natural Language*, (Asilomar, California), 19-22 February 1991.

R. Moore, D. Appelt, J. Bear, M. Dalrymple, and D. Moran, "*SRI's experience with the ATIS evaluation*," in *Proceedings Speech and Natural Language Workshop* held at Hidden Valley, Pennsylvania, pp. 147-148, Morgan Kaufman Publishers, Inc, 24-27 June 1990.

P. Price, V. Abrash, D. Appelt, J. Bear, J. Bernstein, B. Bly, J. Butzberger, M. Cohen, E. Jackson, R. Moore, D. Moran, H. Murveit, and M. Weintraub, "*Spoken language system integration and development*," in *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*, (Kobe, Japan), 1990.

D. B. Moran, "*A case history in computer exploration of fast speech rules*," in *American Journal for Computational Linguistics*, microfiche 19, 1975.

ShopTalk

P. R. Cohen, D. B. Moran, and S. L. Oviatt, "SHOPTALK: An integrated interface for factory information and operation," in *Proceedings of IPC'91*, (Ann Arbor, Michigan), pp. 811-822, Engineering Society of Detroit, April 1991.

P. R. Cohen, M. Dalrymple, D. B. Moran, F. C. N. Pereira, J. W. Sullivan, R. A. Gargan, J. L. Schlossberg, and S. W. Tyler, "*Synergistic use of direct manipulation and natural language*," in *Human Factors in Computing Systems: CHI'89 Conference Proceedings*, (New York, New York), pp. 227-234, ACM, Addison-Wesley Publishing Co., April 1989.

P. R. Cohen, M. Dalrymple, D. B. Moran, and F. C. N. Pereira, "*ShopTalk: an integrated interface for decision support in manufacturing*," in *Working Notes of the AAAI Spring Symposium Series*, vol. AI in Manufacturing, (Stanford University, Stanford, California), pp. 11-15, 28-30 March 1989.

OR-parallelism

H. Alshawi and D. B. Moran, "*The Delphi model and some preliminary experiments*," in *Proceedings of the Fifth International Logic Programming Conference and Symposium*, pp. 1578-1589, The MIT Press, 1988.

Core Language Engine

D. B. Moran and F. C. N. Pereira, "*Quantifier scoping*," in *The Core Language Engine* (H. Alshawi, ed.), ch. 8, pp. 149-172, Cambridge, Massachusetts: A Bradford Book, The MIT Press, 1991.

H. Alshawi, D. M. Carter, J. van Eijck, R. C. Moore, D. B. Moran, and S. G. Pulman, "*Overview of the Core Language Engine*," in *Proceedings of the International Conference on Fifth Generation Computer Systems*, (Tokyo, Japan), pp. 1108-1115, 28 November - 2 December 1988. *http://www.dougmoran.com/dmoran/PAPERS/cle-5thgen88.pdf*

Abstract: The Core Language Engine (CLE) is a domain independent system for translating natural language (English) sentences into formal representations of their literal meanings which are capable of supporting reasoning. It is designed to be used as a major component of interactive advisor systems such as interfaces to database management systems and diagnostic expert systems. The main contribution of the CLE is intended to be substantial coverage of English constructions in both syntax *and* semantics that is well motivated and hence extensible. Interactive facilities are provided to allow users to extend the system vocabulary. The CLE has a modular architecture with well defined interfaces between the various stages of linguistic processing. Unification is used as the mechanism for rule application and passing information in each of morphology, parsing, interpretation, and selectional filtering, so the rules for these components are expressed declaratively. A compact representation of local ambiguities is applied systematically in syntax and semantics, allowing us to adopt the modular staged approach without sacrificing computational efficiency.

D. B. Moran, "*Quantifier scoping in the SRI Core Language Engine*," in *Proceedings of the 26th Annual Meeting of the Association for Computational Linguistics*, (State University of New York at Buffalo), pp. 33-40, 7-10 June 1988.

Command & Control Workstation

E. Craighill, D. Moran, and R. Brungardt, "*Research in information structures and software architectures for C3I systems*," in *Joint Defense Laboratories Conference on Command, Control and Communications (C3)*, (Ft. McNair, Washington, D.C.), June 1987.

L. Aguilar, J. J. Garcia-Luna, D. B. Moran, E. J. Craighill, and R. Brungardt, "*An architecture for a multimedia teleconferencing system*," in *SIGCOMM `86 Symposium: Communications Architectures and Protocols*, August 1986.

A. Poggio, J. J. Garcia-Luna, E. J. Craighill, D. B. Moran, and L. Aguilar, "*CCWS: A computer-based, multimedia information system*," *IEEE Computer*, October 1985.

Montague Grammar

D. B. Moran, "*The representation of inconsistent information in a dynamic model-theoretic semantics*," in *Proceedings of the 20th Annual Meeting of the Association for Computational Linguistics*, (University of Toronto, Toronto, Ontario, Canada), pp. 16-18, 16-18 June 1982.

D. B. Moran, "*Presupposition and implicature in model-theoretic pragmatics*," in *Proceedings of the 19th Annual Meeting of the Association for Computational Linguistics*, (Stanford University, Stanford, California), pp. 107-108, 29 June - 1 July 1981.

D. B. Moran, *Model-Theoretic Pragmatics: Dynamic Models and an Application to Presupposition and Implicature*. PhD thesis, Dept. of Computer and Communication Sciences, The University of Michigan, Ann Arbor, Michigan, October 1980.

J. Friedman, D. B. Moran, and D. S. Warren, "*Evaluating English sentences in a logical model: A process version of Montague grammar*," in *Natural Language Communication with Computers 1980*, proceedings of a workshop conducted as part of an NSF-sponsored US-USSR exchange program, Novosibirsk, USSR, 1979.

J. Friedman, D. B. Moran, and D. S. Warren, "*Evaluating English sentences in a logical model*," in *Information Abstracts, 7th International Conference on Computational Linguistics* (University of Bergen, Norway), 1978, 11pp.

J. Friedman, D. B. Moran, and D. S. Warren, "*Explicit finite intensional models for PTQ*," in *American Journal for Computational Linguistics*, microfiche 74, 1978, pp. 3-22.

J. Friedman, D. B. Moran, and D. S. Warren, "*An interpretation system for Montague grammar*," in *American Journal for Computational Linguistics*, microfiche 74, 1978, pp. 23-96.

Last updated: $Date: 2002/03/18 06:13:13 $

Exhibit 13

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

GOOGLE LLC
Petitioner

v.

IPA TECHNOLOGIES INC.
Patent Owner

Patent No. 6,851,115

DECLARATION OF DR. DOUGLAS B. MORAN

TABLE OF CONTENTS

I, Douglas B. Moran, declare as follows:

I. INTRODUCTION

1. I am of legal age and am competent to make this Declaration.

2. I have been retained by Google LLC as an independent consultant in this proceeding before the United States Patent and Trademark Office ("PTO") regarding U.S. Patent No. 6,851,115 to provide factual information and testimony concerning my prior work and experiences.

3. I am being compensated at my rate of $500 per hour for the time I spend on this matter. My compensation is in no way contingent on the nature of my testimony, or the outcome of this or any other proceeding. I have no other interest in this proceeding.

4. I understand that a copy of my curriculum vitae, which includes a detailed summary of my background, experience, and publications, is provided as Ex. 1008. Below, I describe selected aspects of my background and experience, and one of my joint publications.

II. BACKGROUND

5. I received my Bachelor of Science degree from the Computer Science program in the Department of Electrical Engineering (VI-3) at the Massachusetts Institute of Technology. I subsequently received my M.S. and Ph.D. degrees from the Department of Computer and Communication Sciences at the University of

Michigan (Ann Arbor) with a concentration in Artificial Intelligence and Computational Linguistics. I was also named a Fellow for the A.P. Sloan Foundation grant for Interdisciplinary Studies in Cognitive Sciences in the Department of Linguistics and the Department of Computer and Information Sciences at the University of Massachusetts (Amherst).

6. From 1974 until 1979, I worked as a Teaching Assistant and a Research Assistant in the Department of Computer and Communication Sciences at the University of Michigan. Subsequently, from 1980 until 1983, I worked as an Assistant Professor in the Department of Computer Science at Oregon State University (OSU).

7. My research focus at OSU, as well as my focus in several projects that I worked on subsequently as discussed below, related generally to natural language processing (NLP). The power of using natural language (NL), such as English, instead of formal languages (e.g., designed languages with a formal specification that must be adhered to strictly) for service-related queries and other tasks is that NL utterances can be more concise by dispensing with unnecessary precision, assuming the listener makes correct inferences. My research addressed several issues relating to NLP, including how to implement NLP for improved usability and/or performance in a variety of real-world scenarios and applications.

8. My graduate research was a part of a small group experimenting with computerizing a formal theory of natural language from the area of philosophical logic known as Montague Grammar. It was more sophisticated than the other NLP systems, supporting queries involving time and "possible worlds," for example, the results of multiple runs of a simulator or planning system.

9. At OSU, I supervised the installation of the computer science department's new computer system, which included DEC VAX computer hardware running the UNIX operating system.

10. After my career in academia, I worked on various projects from 1983 until 1999 at SRI International (formerly Stanford Research Institute), a well-known scientific research organization, as a Senior Computer Scientist. My first project involved building distributed systems, and during the course of that project I acquired extensive experience with networked computers. Because the department's computer facility would not support the new Internet workstations (Sun Microsystems workstation computers), I again became involved in system administration, gaining more experience and expertise in the reliable operation of networks of computers and their problems. For the majority of my time at SRI, I worked in the Artificial Intelligence Center (AIC) research group. After leaving SRI, I spent several years working at Recourse Technologies, Inc. (1999-2001) and PacketMotion, Inc. (2003-2004), which were computer security software start-up

companies. I am a named inventor on eight patents relating to my time at Recourse Technologies. From 2004 to 2007, I spent time on civic volunteering, with some technical consulting as well.

11. I have been a consultant for four different legal cases (criminal and civil) involving computer-related issues, and I was an expert witness in one of those cases. I have been retired from my professional career since 2007.

12. Over the course of my career, I have been involved in various professional societies related to artificial intelligence (AI) and computing, including the American Association for Artificial Intelligence (AAAI), the Association for Computational Linguistics (ACL), the Association for Computing Machinery (ACM), the Institute of Electrical and Electronics Engineers (IEEE), and the IEEE Computer Society.

13. I was integral to various research projects at SRI International, including the Open Agent Architecture (OAA) project that I describe below in Section III in more detail. My first project at SRI, lasting from 1983 until 1986, was the Command and Control Workstation (CCWS), which involved the design and implementation of multimedia mail and multimodal conferencing applications. In particular, the CCWS project was starting to develop remote voice conferencing when I joined, and my role was to design and implement what would later be termed an automated agent in the conferencing system that added graphical

information to the conference workspace (e.g., via a map-based display) based on a user's spoken natural language queries to a database. I was the project leader of a team of about 6 to 7 SRI staff members. The CCWS system was funded by the U.S. Navy and was installed on an aircraft carrier and used during an operational military deployment.

14. When I arrived at SRI, the CCWS project was just acquiring UNIX workstations from Sun Microsystems. I took on the role of administrator for a computer network containing those workstations. This network grew substantially when I was assigned responsibility for Sun workstations being used by other groups within the same larger research program. I became more proficient at computer networks based on such experience.

15. One of the drivers of the larger research program that included CCWS was to explore bringing Internet technology to the Navy's communication systems. The underlying role of CCWS was to provide an illustration of capabilities that the Navy's existing communication systems could not support. Having current meteorological data was crucial for many operations, for example using weather to hide ships from opponent's satellites or to know where opponent's ships could be hiding from our satellites. Although the shore-based facilities were capable of providing current meteorological information, the communication system created large gaps between updates. A common complaint was that when this data was

needed, it was often too many hours old to be useful. Consequently, the use of distributed systems was an essential component of CCWS, but the networking software was still only moderately reliable and still maturing. My experience with the problems associated with smaller networks in CCWS was relevant to the subsequent larger, more reliable networks that I later worked on, including for the OAA project that I discuss below in Section III. I also became an expert UNIX programmer in the process of working on the CCWS project.

16. My work on the CCWS project was applicable to so-called "disadvantaged systems," a term that refers to platforms—for example, ships, aircraft, and bases—that do not have state-of-the-art, current technology. This arises from the Navy's long refit cycle (e.g., up to 18 years) and space limitations on board a vessel. For example, a large computer can likely be accommodated on an aircraft carrier, but would be impossible on a smaller vessel or aircraft, due to the computer's space requirements, weight and/or power consumption. For computer applications, the approach was to have a stripped-down version that could be handled by older and less powerful computers, and to have versions that could work with obsolete peripherals. My work on CCWS highlighted the potential for off-loading computation to a remote, more capable facility—a key aspect of distributed computing.

17. Following CCWS, I was recruited to help set up an SRI research lab in Cambridge, England. From 1986 until 1987, I worked on the Core Language Engine (CLE), which was a highly capable natural language processing system that integrated promising new technologies and approaches while applying the lessons learned from earlier large-scale systems. I was a researcher and developer, contributing to the development of the overall system, with primary responsibility for the pragmatics component. I was also responsible for acquiring, setting up, and managing the computer facility in which the CLE system operated. My responsibilities included managing data communications back to SRI's Menlo Park (California) headquarters at a time when affordable connections had intermittent reliably problems.

18. Subsequently, I returned to Menlo Park, transferring to the Artificial Intelligence Center at SRI. From 1987 until 1994, I worked on the Gemini Spoken Language System (SLS), which was based on the CLE system discussed above. The Gemini system re-implemented many of the algorithms from CLE, thereby allowing application of "lessons learned" and design changes to better accommodate integration with the speech recognizer being developed in SRI's Speech Technology and Research (STAR) Laboratory, which was spun off to become Nuance Communications, Inc. For Gemini, I was a researcher and

developer focusing on the integration of speech recognition and natural language understanding technologies.

19. Part of my duties at the Artificial Intelligence Center involved managing the Sun workstations at the computing facility. This grew to about 60 Sun workstations without an increase in staffing because I had heavily automated the maintenance of those systems. The Gemini SLS project required more CPU cycles for running test sets than the budget would allow. However, many of those Sun workstations sat idle much of the time, both during off-hours and when the researcher was off-site. I automated grabbing these available computer resources, using my knowledge of programming and networking.

20. From 1985-1992, I also worked on a project called ShopTalk, for which I was the deputy project leader of a team of about 4 to 8 SRI staff members. ShopTalk was an evolving series of research prototypes that demonstrated the effectiveness of multimodal user interfaces that enabled the user to interact with the system via various input modalities. The user interface provided transparent, integrated access to multiple applications. For example, processing a query could involve combining historical data from a database with projected results from a simulator for various types of scenarios. The ShopTalk project at SRI grew out of a series of collaborations between me and Dr. Philip R. Cohen starting in the Computer Science Department at Oregon State University (OSU). When we both

later landed at SRI, our collaboration resumed. At SRI, I had joined the established Command and Control Workstation (CCWS) project that I discussed above. Prior to his arrival at SRI, Dr. Cohen had been working on the precursor to SRI's ShopTalk system at his former employer, to demonstrate the potential of natural language access to data relating to the microchip fabrication context. When Dr. Cohen arrived at SRI, it was easy for the two of us to combine our two projects (my CCWS system and his ShopTalk system) because they both were built on the same underlying Natural Language Processing (NLP) framework. The CCWS system that I was working on was more developed and had a flashier demo, but it did not have an independent name, so the combined system became known as ShopTalk. Dr. Cohen assumed the title of project leader of ShopTalk and assumed the primary responsibility for funding, and I was system architect and chief implementer, in addition to my role as deputy project leader as discussed above.

21. I made substantial contributions toward creating and customizing presentations and demonstrations relating to the ShopTalk project and presented portions of or all of various demonstrations (presentations showcasing a system's features, often in real-time) to many audiences. Even before I transferred to the Artificial Intelligence Center (AIC), the AIC was using the CCWS version of the system, and me, in demonstrations to visitors and potential clients, for example, then-CEO of Apple John Scully. I had a major role in the creation and evolution of

such presentations, as I watched and listened to the audience and then talked with the engineers while Dr. Cohen talked to the executives. In a typical debriefing that occurred after a demonstration, we revised the system in large and small ways.

22. Another important part of my background was my experience with various difficulties with accessing databases. For projects researching NLP access to databases, the U.S. Navy had provided the "Blue File," a small, sanitized (unclassified) version of an actual database. As I experienced in the 1980s, this and other realistic databases highlighted that natural language (NL) access to databases was often complicated by inconsistencies between a database's documented description and its actual contents. My first brush with the difficulty of integrating information from multiple databases came as a summer intern at Corning Glass Works (now Corning Inc.) in 1971. While core functions had already been computerized, there were many "databases" that were paper forms in filing cabinets. A junior programmer and I were assigned the first step of finding out what was actually in the paper databases, at the level a programmer would need to understand. I was surprised to find that the real (paper) database bore only a faint resemblance to its documented description. There were workarounds, traditions, extensions, and entries whose interpretation depended on the user knowing the context. Such issues with existing databases informed my work on later projects.

III. OPEN AGENT ARCHITECTURE (OAA)

23. Beginning in 1994, I worked on the Open Agent Architecture (OAA) project, which was a distributed multi-agent system focused on rapid assembly of new distributed systems from existing software applications. The OAA project focused on facilitating the creation of systems from components that were not intended to work together by providing a higher level of abstraction than that found in other technologies such as objected-oriented programming (OOP) methodologies and Object Request Brokers (ORB). The OAA distributed computing system included a multimodal user interface enabling the user to interact with the system via speech (e.g., natural language voice), gestures, and a graphical user interface (GUI), including GUI elements such as windows, icons, menus, and a pointer. The OAA system was inspired by our experiences with the ShopTalk project (discussed above in Section II).

24. Dr. Cohen and I had different interests in the Shoptalk and OAA systems. He had strong interests in researching multimodal interactions. My interest was in using multimodal interfaces to enhance Natural Language Processing (NLP) systems and permit users to supplement use of NLP in various real-world scenarios.

25. Lessons that I had learned from my experiences with databases (*see* my discussion above in Section II) were an important component of the OAA

design, particularly the lesson that knowledge about the database outside of its formal specification should not be built into the core OAA system—the facilitator and the user interface agents—but rather should be placed in the specific database agent. I conceived of that agent likely being a subsidiary facilitator with multiple agents: There would be an agent accessing the database in its own query language and then one or more agents "cleaning up" those results into forms useful at the next level up. This cleanup could be more than simple transformations, such as a response to the initial query requesting disambiguation.

26. My management of computer facilities (*see* my discussion above in Section II) impressed upon me the importance of computer security. The culture of most distributed systems in AI research at the time was one of "total trust," where agents were assumed to be neither malicious nor malfunctioning when those systems were being evaluated. The designs of many of the "Intelligent Agent" systems of other research groups appeared to impede, and possibly preclude, dealing with a range of security issues. Although the OAA system did not have the funding to include security features, I included these concerns in its evolving design.

27. I became the OAA project leader when Dr. Cohen left SRI in August 1994. With that change in leadership came a shift in the focus of the project. The multimodal interface was no longer a research focus, although those components

were improved and alternatives were added to the system. For example, we added the Gemini NLP system—on which I had worked—to the OAA system.

28. The facilitator was at the core of the OAA system, and I saw having more agents as the best way to drive its development and to evaluate it. Some of these agents would be complementary—for example, agents handling different parts of what was conceptually a single database to the user.

29. The OAA architecture was different from those in a number of other intelligent agent research projects of that time that used a "blackboard" approach (where multiple processes communicate by accessing and modifying data from a global data store) or similar architectures. I believed that such approaches could not easily be expanded to support large numbers of agents. To provide responsiveness, the agents in the blackboard approach needed to frequently poll the blackboard to see if there were any tasks waiting for them. This consumed lots of CPU time and occupied working memory. UNIX networking had encountered the same problem much earlier and the solution was to have a single lightweight program—a dispatcher—listening for network service requests and then starting up the appropriate program and giving the request to it. Such a solution also simplified those network service programs because they did not have to manage a queue of requests; instead, for each request, the dispatcher could create its own copy of the service program. The dispatcher was initialized with a configuration

file (small database) describing the service programs that it was handling. The OAA facilitator that my team at SRI developed has the same basic architecture, but in addition to being a dispatcher, it decomposed complex requests into simpler ones and managed communications between the agents, so that as part of processing a request, an agent could make requests of other agents.

30. My prior experiences in the area of networking (discussed above in Section II) influenced my focus for the OAA facilitator and remote agents. Since many AI systems are compute-intensive, I saw it as important to be able to use resources spread over many computers, particularly in light of my experiences and insights regarding "disadvantaged systems" as I discussed above in Section II. I was acutely aware of the proclivity of those computers to freeze up or crash, so being able to have redundancy was important. For example, I recognized the need for having multiple computers with copies of the same agent, with the facilitator able to select which one(s) to use for servicing a particular task. Additionally, I expected that one would want to have multiple related agents on the same remote computer. It was helpful if those agents did not have to communicate with each other through the double bottleneck of the network and a centralized facilitator. Consequently, it was beneficial to have subsidiary facilitators on remote computers that would coordinate the activities of the agents there.

31. I also wanted subsidiary facilitators from the perspective of software engineering and formal systems. Recursion is an important concept in computer science and is basic to AI, and especially NLP systems, as it allows regularities (e.g., regularly occurring structures, phenomena, or patterns) to be captured in a complex system. By enabling a simpler and smaller description, the number of errors can be reduced. For example, a simple sentence is a single clause and a complex sentence contains a main clause and one or more subsidiary clauses. Moreover, a subsidiary clause can have its own main clause and subsidiary clauses. Implementing a recursive approach for processing such clauses can be more efficient and accurate than using a non-recursive approach. I saw having a control structure that embodied recursion as the best way to support easy scaling up to more agents running on more computers.

32. For OAA, I substantially rewrote Dr. Cohen's presentation that I discussed above in Section II, and I refined the demonstration scenario. The OAA demonstration became very popular with SRI's Business Development staff and with SRI executives as part of marketing SRI's general capabilities. For each potential client, the presentation and demonstration needed to be customized—anywhere from cosmetic changes to tweaks to substantial enhancements. The frequency of the demonstrations constrained what needed to be developed when,

but I also had latitude in which customizations to make and used these choices to drive the direction of the OAA project.

33. In 1995 a nerve injury in my neck and shoulder reduced the hours I could spend using a keyboard. The intensity of the problem varied over the next years. As a result of my injury, I did little programming for OAA after 1995 because programming was the lowest priority of the project tasks for me. In addition to having primary responsibility for the presentations, demonstration scenarios, funding proposals and research publications, I also conducted code reviews and design sessions. Because the OAA project team was small, most of these occurred in offices. Because the team was co-located and small, many of these meetings were on-demand: If a team member had something that needed discussion, it was usually easy for the relevant members to gather on very short notice. Additionally, there was a whiteboard in the hall outside my office, so members of the OAA team often gathered in that common area.

34. The demands of another project that I was working on in the area of computer security reduced my role in OAA, and I left the team around 1998. I left SRI in May 1999 for a computer security start-up.

IV. THE OAA PAPER

35. While I was at SRI, two of my colleagues on the OAA project (David Martin and Adam Cheyer) and I co-authored a paper entitled "Building Distributed Software Systems with the Open Agent Architecture" (I will refer to this paper as "the OAA paper" for short). My understanding is that the OAA paper was presented by one of my co-authors at the Third International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology, March 23-25 1998, London, UK ("PAAM98"). Although I could not attend that conference due to my neck/shoulder injury (discussed above), it was customary for artificial intelligence conferences such as PAAM98 that papers submitted for such conferences were expected to be presented by one of the co-authors. It was also common for such papers to be made available on the Web by the authors and sometimes by the conference itself, though I do not remember if the PAAM98 conference made the OAA paper available on the Web. Artificial intelligence conferences such as PAAM98 were generally open to any member of the public willing to attend, including professionals in the area of networked computer systems and those otherwise interested or performing research in the field to whom such conferences were publicized and many of whom commonly attended in person. Proceedings from such conferences were typically distributed in printed/hardcopy format (including the text of papers accepted by the conferences)

to attendees who paid the registration fees for the conferences. I understand that a copy of the published OAA paper from the conference proceedings of the PAAM98 conference is submitted as Exhibit 1011 in this proceeding, and I have reviewed that paper for this proceeding and recognize it.

36. The process for developing the OAA system generally, and for preparing papers related to OAA, typically involved substantial discussion between the team members. I typically participated in developing ideas for features based on such discussions. I often prepared a demonstration of technical features and ideas and presented many demonstrations to various audiences, as I mentioned above in Section II in the context of demonstrations at SRI generally. I frequently adapted what I verbalized in demonstrations for incorporation into papers (such as the OAA paper).

37. In computer science, AI, and many other scientific fields, the convention at the time of the OAA paper was (and still is) that authorship of a paper is accorded to all who made significant contributions to the work described, and not just to those who wrote the paper. If there are people who made smaller, but still noteworthy, contributions, they are listed in a section typically called "Acknowledgments". The order in which the author is listed is highly variable. Sometimes the most senior person is listed first, sometimes it is the person who will be presenting the paper, sometimes it is the person who was the primary

author, and sometimes it is the person with the largest contribution to the work, with alphabetic order used when there are problems with this. And sometimes, the first listed author is the person who would benefit the most from that position. Much of the writing in the OAA paper is mine, either directly borrowed from or derived from papers, presentations and proposals that I wrote.

38. Many computer science and AI conferences around the time of PAAM98 had a paper submission deadline 6 to 8 months in advance of the conference, to accommodate reviewing, selection by the conference program committee, revisions in the accepted papers and printing. The submission deadline for PAAM98—held in March 1998—was likely September 1997. Being an international conference, the deadline may well have been 1-3 months earlier than that. Before a paper could be submitted, it needed to pass through SRI's professional editors and be approved at the Vice President level. Based on these factors, I would estimate that my co-authors and I likely began preparation of the OAA paper in late spring or early summer of 1997, if not earlier.

39. The OAA paper relates to various aspects of a networked, distributed computing system and to using the Open Agent Architecture (OAA) to build distributed software systems, and the paper covers a lot of my contributions. For example, I played a major role in various networking-related concepts described in the OAA paper, as they built on my knowledge and experiences from having

worked on the CCWS project (as I discussed above in Section II) and also built on my other knowledge and expertise in the areas of distributed systems and networking. Based on my experience operating a computer facility at SRI and working in computer security as discussed above in Section II, I had substantial experience in distributed systems, and contributed to the conception of distributed technologies that are at the core of the OAA paper, e.g., as described at Sections 2.5 and 3 of the OAA paper. (Ex. 1011 at 358-62.) In particular, I played a significant role regarding the distributed agent-based approach, and in particular using a facilitator, as described in the OAA paper at Section 4 and 4.1-4.5. (*Id.* at 362-69.) I also played a significant role in the approach of using recursion to decompose base goals into subgoals that were then dispatched to agents, e.g., as described in the OAA paper at Sections 2.5 and 4.1-4.2. As discussed above, e.g., in Sections II and III, my personal experiences and background helped me conceive of and add such contributions to the OAA paper.

V. CONCLUSION

40. I declare that all statements made herein of my knowledge are true, and that all statements made on belief are believed to be true, and that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code.

Dated: Feb 24, 2019 By: Dr. Douglas B Moran
Dr. Douglas B. Moran

Exhibit 14
SEALED

REDACTED IN ITS ENTIRETY

Exhibit 15

OAA Distribution and Documentation

Distribution

As with many beta programs, OAA 1.0 distribution has been limited to a selected number of users.

OAA 2.0 will be open and freely distributed for non-commercial use on this web site in 2Q97.
Read the official statement on OAA distribution here.

Documentation

Documentation for OAA v2.0 will be placed here when completed.

This will include:

- Programmer's Guide
- API Reference Manual
- Tutorial

Active OAA participants may enter here.

Exhibit 16

OAA-USERS Mailing List Archive by date

- **Most recent messages**
- **Messages sorted by:** [thread][subject][author]
- **Other mail archives**

Starting: *Fri 21 Jun 1996 - 00:00:-30276 PDT*
Ending: *Thu 26 Aug 1999 - 17:01:06 PDT*
Messages: 40

- **OAA-USERS mailing list** *Doug Moran*
- **bug: srx agent on standalone machine** *David Martin*
- **setup.pl in common directory** *David Martin*
- **FIPA report** *Adam Cheyer*
- **dynamic language models** *John Dowding*
- **problem with passing floats in OAA** *John Dowding*
- **passing floating point numbers through library(tcp)** *John Dowding*
- **Re: dynamic language models** *David Martin*
- **john makes a mistake** *John Dowding*
- **StartIt: onready, ondisconnect** *Adam Cheyer*
- **Alert: XWindows agents** *Adam Cheyer*
- **Release Version** *Adam Cheyer*
- **Trigger Changes** *Adam Cheyer*
- **New event scheme** *Adam Cheyer*
- **Re: New event scheme** *Luc JULIA*
- **Re: New event scheme** *David Martin*
- **Re: New event scheme** *Adam Cheyer*
- **Proposal: New Data management** *Adam Cheyer*
- **Re: Proposal: New Data management** *David Martin*
- **PC News** *Adam Cheyer*
- **C Library: structured message proposal** *Adam Cheyer*
- **Re: C Library: structured message proposal** *David Martin*
- **Re: C Library: structured message proposal** *Victor S. Abrash*
- **Re[2]: C Library: structured message proposal** *Martin Fong*
- **Re[3]: C Library: structured message proposal** *Keith Skinner*
- **oaa-users mailing list** *Adam Cheyer*
- **OAA Mailing List** *The Emminizer's*
- **Strings in the OAA C library** *Adam Cheyer*
- **Re: Strings in the OAA C library** *Ralph Becket*
- **Inquiry concerning NLP API** *Sharon Settnek*
- **Re: Inquiry concerning NLP API** *Adam Cheyer*
- **Loopy Triggers** *Stuart Lowry*
- **Re: Loopy Triggers** *Adam Cheyer*
- **writeBB <-> readBB question** *Stuart Lowry*
- **Re: writeBB <-> readBB question** *Adam Cheyer*
- **New facilitator prompting for port** *John Dowding*
- **OAA library for Perl** *John Dowding*
- **Re: OAA library for Perl** *Luc JULIA*
- **Large data sets** *Stuart Lowry*
- **Re: Large data sets** *Adam Cheyer*

Last message date: *Thu 26 Aug 1999 - 17:01:06 PDT*

Exhibit 31

Archived on: *Thu Aug 26 1999 - 17:01:12 PDT*

- **Messages sorted by:** [thread][subject][author]
- **Other mail archives**

This archive was generated by hypermail 1.02.

OAA-USERS Mailing List Archive by thread

- **Most recent messages**
- **Messages sorted by:** [date][subject][author]
- **Other mail archives**

Starting: *Fri 21 Jun 1996 - 00:00:-30276 PDT*
Ending: *Thu 26 Aug 1999 - 17:01:06 PDT*
Messages: 40

- **OAA-USERS mailing list** *Doug Moran*
- **bug: srx agent on standalone machine** *David Martin*
- **setup.pl in common directory** *David Martin*
- **FIPA report** *Adam Cheyer*
- **dynamic language models** *John Dowding*
    - **Re: dynamic language models** *David Martin*
- **problem with passing floats in OAA** *John Dowding*
- **passing floating point numbers through library(tcp)** *John Dowding*
- **john makes a mistake** *John Dowding*
- **StartIt: onready, ondisconnect** *Adam Cheyer*
- **Alert: XWindows agents** *Adam Cheyer*
- **Release Version** *Adam Cheyer*
- **Trigger Changes** *Adam Cheyer*
- **New event scheme** *Adam Cheyer*
    - **Re: New event scheme** *Luc JULIA*
    - **Re: New event scheme** *David Martin*
    - **Re: New event scheme** *Adam Cheyer*
- **Proposal: New Data management** *Adam Cheyer*
    - **Re: Proposal: New Data management** *David Martin*
- **PC News** *Adam Cheyer*
- **C Library: structured message proposal** *Adam Cheyer*
    - **Re: C Library: structured message proposal** *Victor S. Abrash*
- **Re: C Library: structured message proposal** *David Martin*
- **Re[2]: C Library: structured message proposal** *Martin Fong*
- **Re[3]: C Library: structured message proposal** *Keith Skinner*
- **oaa-users mailing list** *Adam Cheyer*
- **OAA Mailing List** *The Emminizer's*
- **Strings in the OAA C library** *Adam Cheyer*
- **Re: Strings in the OAA C library** *Ralph Becket*
- **Inquiry concerning NLP API** *Sharon Settnek*
    - **Re: Inquiry concerning NLP API** *Adam Cheyer*
- **Loopy Triggers** *Stuart Lowry*
    - **Re: Loopy Triggers** *Adam Cheyer*
- **writeBB <-> readBB question** *Stuart Lowry*
    - **Re: writeBB <-> readBB question** *Adam Cheyer*
- **New facilitator prompting for port** *John Dowding*
- **OAA library for Perl** *John Dowding*
    - **Re: OAA library for Perl** *Luc JULIA*
- **Large data sets** *Stuart Lowry*
    - **Re: Large data sets** *Adam Cheyer*

Last message date: *Thu 26 Aug 1999 - 17:01:06 PDT*

Archived on: *Thu Aug 26 1999 - 17:01:12 PDT*

- **Messages sorted by:** [date][subject][author]
- **Other mail archives**

This archive was generated by hypermail 1.02.

OAA-USERS Mailing List Archive by subject

- **Messages sorted by:** [date][thread][author]
- **Other mail archives**

Starting: *Fri 21 Jun 1996 - 00:00:-30276 PDT*
Ending: *Thu 26 Aug 1999 - 17:01:06 PDT*
Messages: 40

- **Alert: XWindows agents**
  - Adam Cheyer *Thu, 03 Oct 1996 11:44:48 -0700*
- **bug: srx agent on standalone machine**
  - David Martin *Thu, 27 Jun 1996 16:37:05 -0700 (PDT)*
- **C Library: structured message proposal**
  - Victor S. Abrash *Thu, 22 May 1997 19:09:34 -0700*
  - David Martin *Thu, 22 May 1997 17:33:07 -0700 (PDT)*
  - Adam Cheyer *Thu, 22 May 1997 16:56:47 -0700*
- **dynamic language models**
  - David Martin *Tue, 23 Jul 1996 20:43:31 +0000*
  - John Dowding *Mon, 22 Jul 1996 10:49:22 -0700 (PDT)*
- **FIPA report**
  - Adam Cheyer *Mon, 01 Jul 1996 18:40:32 -0700*
- **Inquiry concerning NLP API**
  - Adam Cheyer *Tue, 03 Nov 1998 18:57:51 -0800*
  - Sharon Settnek *Tue, 03 Nov 1998 10:23:36 -0500*
- **john makes a mistake**
  - John Dowding *Wed, 31 Jul 1996 18:36:51 -0700 (PDT)*
- **Large data sets**
  - Adam Cheyer *Thu, 26 Aug 1999 17:01:01 -0700*
  - Stuart Lowry *Wed, 7 Jul 1999 08:40:44 -0400 (EDT)*
- **Loopy Triggers**
  - Adam Cheyer *Thu, 03 Dec 1998 08:24:08 -0800*
  - Stuart Lowry *Wed, 02 Dec 1998 15:20:07 -0500*
- **New event scheme**
  - Adam Cheyer *Wed, 23 Oct 1996 16:28:49 -0700*
  - David Martin *Mon, 21 Oct 1996 17:54:48 +0000*
  - Luc JULIA *Sun, 20 Oct 1996 17:56:17 -0700 (PDT)*
  - Adam Cheyer *Sun, 20 Oct 1996 17:34:25 -0700*
- **New facilitator prompting for port**
  - John Dowding *Thu, 22 Apr 1999 10:33:36 -0700*
- **OAA library for Perl**
  - Luc JULIA *Tue, 06 Jul 1999 20:09:49 -0700*
  - John Dowding *Tue, 06 Jul 1999 18:35:36 -0700*
- **OAA Mailing List**
  - The Emminizer's *Fri, 23 Oct 1998 19:04:15 -0400*
- **oaa-users mailing list**
  - Adam Cheyer *Mon, 19 Oct 1998 18:33:43 -0700*
  - Doug Moran *Fri, 21 Jun 1996 15:35:24 -0700 (PDT)*
- **passing floating point numbers through library(tcp)**
  - John Dowding *Mon, 22 Jul 1996 11:09:27 -0700 (PDT)*
- **PC News**
  - Adam Cheyer *Tue, 05 Nov 1996 16:53:50 -0800*

- **problem with passing floats in OAA**
  - John Dowding *Mon, 22 Jul 1996 10:56:35 -0700 (PDT)*
- **Proposal: New Data management**
  - David Martin *Thu, 24 Oct 1996 18:26:45 +0000*
  - Adam Cheyer *Thu, 24 Oct 1996 10:15:42 -0700*
- **Re[2]: C Library: structured message proposal**
  - Martin Fong *Thu, 22 May 1997 19:29:37 -0700*
- **Re[3]: C Library: structured message proposal**
  - Keith Skinner *Tue, 27 May 97 09:51:08 PDT*
- **Release Version**
  - Adam Cheyer *Thu, 17 Oct 1996 08:05:42 -0700*
- **setup.pl in common directory**
  - David Martin *Thu, 27 Jun 1996 18:30:59 -0700 (PDT)*
- **StartIt: onready, ondisconnect**
  - Adam Cheyer *Thu, 01 Aug 1996 17:26:08 -0700*
- **Strings in the OAA C library**
  - Ralph Becket *Fri, 30 Oct 1998 15:26:49 +0000*
  - Adam Cheyer *Fri, 30 Oct 1998 06:23:17 -0800*
- **Trigger Changes**
  - Adam Cheyer *Thu, 17 Oct 1996 08:39:46 -0700*
- **writeBB <-> readBB question**
  - Adam Cheyer *Thu, 17 Dec 1998 08:52:16 -0800*
  - Stuart Lowry *Wed, 16 Dec 1998 09:35:15 -0500*

Last message date: *Thu 26 Aug 1999 - 17:01:06 PDT*
Archived on: *Thu Aug 26 1999 - 17:01:13 PDT*

- **Messages sorted by:** [date][thread][author]
- **Other mail archives**

This archive was generated by hypermail 1.02.

OAA-USERS Mailing List Archive by author

- **Messages sorted by:** [date][thread][subject]
- **Other mail archives**

Starting: *Fri 21 Jun 1996 - 00:00:-30276 PDT*
Ending: *Thu 26 Aug 1999 - 17:01:06 PDT*
Messages: 40

- **Adam Cheyer**
    - Re: Large data sets *Thu, 26 Aug 1999 17:01:01 -0700*
    - Re: writeBB <-> readBB question *Thu, 17 Dec 1998 08:52:16 -0800*
    - Re: Loopy Triggers *Thu, 03 Dec 1998 08:24:08 -0800*
    - Re: Inquiry concerning NLP API *Tue, 03 Nov 1998 18:57:51 -0800*
    - Strings in the OAA C library *Fri, 30 Oct 1998 06:23:17 -0800*
    - oaa-users mailing list *Mon, 19 Oct 1998 18:33:43 -0700*
    - C Library: structured message proposal *Thu, 22 May 1997 16:56:47 -0700*
    - PC News *Tue, 05 Nov 1996 16:53:50 -0800*
    - Proposal: New Data management *Thu, 24 Oct 1996 10:15:42 -0700*
    - Re: New event scheme *Wed, 23 Oct 1996 16:28:49 -0700*
    - New event scheme *Sun, 20 Oct 1996 17:34:25 -0700*
    - Trigger Changes *Thu, 17 Oct 1996 08:39:46 -0700*
    - Release Version *Thu, 17 Oct 1996 08:05:42 -0700*
    - Alert: XWindows agents *Thu, 03 Oct 1996 11:44:48 -0700*
    - StartIt: onready, ondisconnect *Thu, 01 Aug 1996 17:26:08 -0700*
    - FIPA report *Mon, 01 Jul 1996 18:40:32 -0700*
- **David Martin**
    - Re: C Library: structured message proposal *Thu, 22 May 1997 17:33:07 -0700 (PDT)*
    - Re: Proposal: New Data management *Thu, 24 Oct 1996 18:26:45 +0000*
    - Re: New event scheme *Mon, 21 Oct 1996 17:54:48 +0000*
    - Re: dynamic language models *Tue, 23 Jul 1996 20:43:31 +0000*
    - setup.pl in common directory *Thu, 27 Jun 1996 18:30:59 -0700 (PDT)*
    - bug: srx agent on standalone machine *Thu, 27 Jun 1996 16:37:05 -0700 (PDT)*
- **Doug Moran**
    - OAA-USERS mailing list *Fri, 21 Jun 1996 15:35:24 -0700 (PDT)*
- **John Dowding**
    - OAA library for Perl *Tue, 06 Jul 1999 18:35:36 -0700*
    - New facilitator prompting for port *Thu, 22 Apr 1999 10:33:36 -0700*
    - john makes a mistake *Wed, 31 Jul 1996 18:36:51 -0700 (PDT)*
    - passing floating point numbers through library(tcp) *Mon, 22 Jul 1996 11:09:27 -0700 (PDT)*
    - problem with passing floats in OAA *Mon, 22 Jul 1996 10:56:35 -0700 (PDT)*
    - dynamic language models *Mon, 22 Jul 1996 10:49:22 -0700 (PDT)*
- **Keith Skinner**
    - Re[3]: C Library: structured message proposal *Tue, 27 May 97 09:51:08 PDT*
- **Luc JULIA**
    - Re: OAA library for Perl *Tue, 06 Jul 1999 20:09:49 -0700*
    - Re: New event scheme *Sun, 20 Oct 1996 17:56:17 -0700 (PDT)*
- **Martin Fong**
    - Re[2]: C Library: structured message proposal *Thu, 22 May 1997 19:29:37 -0700*
- **Ralph Becket**
    - Re: Strings in the OAA C library *Fri, 30 Oct 1998 15:26:49 +0000*
- **Sharon Settnek**

- Inquiry concerning NLP API *Tue, 03 Nov 1998 10:23:36 -0500*
- **Stuart Lowry**
  - Large data sets *Wed, 7 Jul 1999 08:40:44 -0400 (EDT)*
  - writeBB <-> readBB question *Wed, 16 Dec 1998 09:35:15 -0500*
  - Loopy Triggers *Wed, 02 Dec 1998 15:20:07 -0500*
- **The Emminizer's**
  - OAA Mailing List *Fri, 23 Oct 1998 19:04:15 -0400*
- **Victor S. Abrash**
  - Re: C Library: structured message proposal *Thu, 22 May 1997 19:09:34 -0700*

Last message date: *Thu 26 Aug 1999 - 17:01:06 PDT*
Archived on: *Thu Aug 26 1999 - 17:01:13 PDT*

- **Messages sorted by:** [date][thread][subject]
- **Other mail archives**

This archive was generated by hypermail 1.02.

OAA-USERS mailing list

Doug Moran (*(no email)*)
Fri, 21 Jun 1996 15:35:24 -0700 (PDT)

- **Messages sorted by:** [date][thread][subject][author]
- **Next message:** David Martin: "bug: srx agent on standalone machine"

All:

We have created a mailing list for discussion among the users of OAA.
The purpose of this list is to allow users to share experience, ask
questions of the larger group, and make suggestions for enhancements
that other users can comment on.

The mailing address of this list is:
oaa-users@ai.sri.com
And an HTML-based archive of messages is accessible at:
<A HREF="http://www.ai.sri.com/~oaa/distribution/MailArchive/oaa-users/index.html">OAA Users archive</A>

Membership:
This list will include
1. the OAA project staff
2. SRI personnel using OAA on other projects
3. outside users of OAA.

Note: messages to this list will be seen by all these other people, not
just the OAA project staff.

To send a message to just the OAA project staff (that other users will not
see), please direct it to the appropriate individuals.

** PLEASE respond if you wish to be included on this list***
If there are colleagues that should be included, please send me a note
and I will add them.
I would also appreciate a response if you do not wish to be included
so that I know that all of you have received this message.

Possible reason for not being on the list:
The sendmail "expn" command shows the membership of this list, and you
might not want to be listed.

Even if you are not on the mailing list, you can check the archives
(the archives are currently in the private portion of the OAA web site
to which SRI personnel and designated SRI clients have access).

-- Doug Moran

- **Next message:** David Martin: "bug: srx agent on standalone machine"

bug: srx agent on standalone machine

David Martin (*(no email)*)
Thu, 27 Jun 1996 16:37:05 -0700 (PDT)

- **Messages sorted by:** [date][thread][subject][author]
- **Next message:** David Martin: "setup.pl in common directory"
- **Previous message:** Doug Moran: "OAA-USERS mailing list"

oaa-users -

Recently 2 different OAA user groups uncovered a problem with the "srx" (Speech Recognition) agent, which occurs ONLY when an agent system is being run on a standalone machine (that is, detached from the network).

The problem is that srx is not able to establish communications with the Nuance (or Corona) recognition server. The symptom is an error message from srx that says:
NUANCE_SERVER_NOT_ACCESSIBLE
or
CORONA_SERVER_NOT_ACCESSIBLE

The problem turned out to be quite hard to diagnose, but it now appears that it's due to a nonstandard C library (libc.a) with which srx was linked. This is a nonstandard library that was created for use at SRI, and which changes the normal behavior in resolving host names.

This same nonstandard library could conceivably affect other agents compiled at SRI, but as far as we know, only srx has exhibited a problem.

There appears to be a simple workaround (although it has not yet been tested very extensively):

Add the following line to the file
/etc/resolv.conf
on the standalone machine:

order local,bind

On our machines here at SRI, it is the first non-commented line, so it might be wise to place it in that position.

- Dave Martin

- **Next message:** David Martin: "setup.pl in common directory"
- **Previous message:** Doug Moran: "OAA-USERS mailing list"

setup.pl in common directory

David Martin (*(no email)*)
Thu, 27 Jun 1996 18:30:59 -0700 (PDT)

- **Messages sorted by:** [date][thread][subject][author]
- **Next message:** Adam Cheyer: "FIPA report"
- **Previous message:** David Martin: "bug: srx agent on standalone machine"

OAA developers -

Someone (I assume someone from the Speech group - the user id appears to me as "1043"), created the file
/home/trestle4/OAA/demo/setup.pl
(I suppose the path may appear differently to users on speech machines.)

I found it necessary to remove this file (actually I moved it to setup.pl.sav), because another user (Elizabeth) was simultaneously trying to run some of the agents in this directory.

When an agent starts up, if there is a setup.pl file in the current directory, it will take precedence over ~/setup.pl. So Elizabeth's agents were trying to connect to the host/port specified in ./setup.pl. Because the current directories for her agents are specified in startit config files, it's too much trouble to try to change her setup.

In general, then, please do not place a "setup.pl" file in
/home/trestle4/OAA/demo
or any other agent directories that are intended for common use.

Thanks!

- Dave Martin

- **Next message:** Adam Cheyer: "FIPA report"
- **Previous message:** David Martin: "bug: srx agent on standalone machine"

FIPA report

Adam Cheyer (*cheyer@ai.sri.com*)
Mon, 01 Jul 1996 18:40:32 -0700

- **Messages sorted by:** [date][thread][subject][author]
- **Next message:** John Dowding: "dynamic language models"
- **Previous message:** David Martin: "setup.pl in common directory"

Well, I'm back from the 2nd FIPA (Foundation for Intelligent Physical Agents) meeting in Yorktown, NY. FIPA is an effort by the instigator of the MPEG consortium to standardize agent technology. So here's the report...

In general, the workshop started out quite disorganized, basically because so many different points of view were represented: some people were interested in robots (about 10% only), some about multimedia diffusion (video on demand) and learning user's preferences to select appropriate content, some interested in mobile, general magic style agents, and others were more in the distributed software agents area. Since no one clear vision was presented of what an agent is, much discussion was spent on what would be standardized and what wouldn't.

The second day was a little more efficient, as questions were prohibited during the presentations. We were able to see the broad spectrum of requirements for all the different participants. There were many very interesting presentations condensed into a very short time. We will receive (in a few weeks) slides from all the presentations; I can send you copies if you like.

Of the presentations I saw, my favorites were :

Donald Steiner, Siemens : their MECCA architecture shares many qualities with OAA. They are applying the technology to the domains of office assistant (very similar to OAA!) and to trip planning.

Albert D. Baker (University of Cincinnati): real world domain using a broadcast mechanism to make 1000 (!) agents interact.

Nader Azarmi (BT Labs): very interesting negotiation-capable agents applied to the domain of business process control.

For my brief presentation, I tried to stress the following points:

1. In my opinion, it is important for FIPA to carefully define what is an agent and how this differs from other technologies such as objects. What are the requirements of an agent architecture vs. a distributed object architecture? Can't we reuse many

common notions (such as security) that standards like CORBA already address?

2. At SRI, we have one possible definition of what an agent should be, focusing strongly on distributed, delegated "intelligence" and control, and on a high-level communication language.

3. We've implemented a number of applications within the framework (probably more than any other group I saw at FIPA) and with this experience, we've learned some valuable lessons, one being the utility of providing hooks for extensible tools to simplify interfacing with and using agents.

In the third day, some progress was made in defining a list of requirements, target applications, and standardisation items to be considered. "Results" can be seen on the FIPA homepage, at http://www.cselt.stet.it/ufv/leonardo/fipa.htm.

I think that given all the different viewpoints, FIPA seems to be moving towards creating some sort of standard that will primarily reflect a distributed software agents perspective. Everything will become much more concrete during the next meeting in October, which is being held in Japan: at this time, a call for proposals will be issued. I think that, if pushed by some tangible applications which focus the vision a little, FIPA might actually arrive at a defining a useful set of standards that will allow agent interoperation on a larger scale than currently available. The intent is to have the first FIPA standards defined by the end of 1997...

One question is, what is the role of OAA in all this? I believe that if we choose to, we are in a strong position to influence much of what FIPA produces as a standard. I think that perhaps with the Siemens and BT Labs, we really have the closest overall framework and approach for what FIPA is trying to accomplish. And we are the only group who is trying to use their framework to implement the complete range of agent applications, from cooperating robots, to intelligent avatars, to multimodal and multimedia interfaces, to information retrieval and management. The release of a distributable version of OAA-lite should only enhance our bargaining power.

So, that's the report. If you have questions, feel free to ask...

Adam.

- **Next message:** John Dowding: "dynamic language models"
- **Previous message:** David Martin: "setup.pl in common directory"

dynamic language models

John Dowding (*(no email)*)
Mon, 22 Jul 1996 10:49:22 -0700 (PDT)

- **Messages sorted by:** [date][thread][subject][author]
- **Next message:** John Dowding: "problem with passing floats in OAA"
- **Previous message:** Adam Cheyer: "FIPA report"
- **Next in thread:** David Martin: "Re: dynamic language models"

In the near term for CommandTalk, I want to add to the capability to dynamically update the NL and recognition grammars with new open-class lexical items. I'd like to do this in a way that is consistent with the way it has been done on other OAA projects. So, I have two questions:

- what is the format for the messages that are currently used to do this?

- is the code for the text-to-phone agent available?

Given the right message format, I will update the Gemini-based NL agent to solve those messages.

Are there any other gotchas that I should be aware of?

John

- **Next message:** John Dowding: "problem with passing floats in OAA"
- **Previous message:** Adam Cheyer: "FIPA report"
- **Next in thread:** David Martin: "Re: dynamic language models"

problem with passing floats in OAA

John Dowding (*(no email)*)
Mon, 22 Jul 1996 10:56:35 -0700 (PDT)

- **Messages sorted by:** [date][thread][subject][author]
- **Next message:** John Dowding: "passing floating point numbers through library(tcp)"
- **Previous message:** John Dowding: "dynamic language models"

I tracked down a bug in the OAA with passing floats in agent messages. The problem is that when a goal is solved, and the answer is passed back to the facilitator, the response will not unify with the call. This is because floats will only unify if the internal representation is bit-identical, but this is not happening when the floats are recreated with prolog's read().

I have settled on a work-around of quoting all floats in CommandTalk, then doing a remove_quotes() prior to converting them to floats in C. This suggestion came from Didier.

I am also going to post a note to hotline@quintus.com, because I thought that they choose to use exponential notation when printing floats to solve exactly this problem.

- **Next message:** John Dowding: "passing floating point numbers through library(tcp)"
- **Previous message:** John Dowding: "dynamic language models"

passing floating point numbers through library(tcp)

John Dowding (*(no email)*)
Mon, 22 Jul 1996 11:09:27 -0700 (PDT)

- **Messages sorted by:** [date][thread][subject][author]
- **Next message:** David Martin: "Re: dynamic language models"
- **Previous message:** John Dowding: "problem with passing floats in OAA"

We are using library(tcp) to pass prolog terms between prolog processess. Since I am passing goals from one process to another, I need the (further instantiated) term to unify with the goal when it is passed back. There is a problem when the goal contains a floating point number. In this case, the returning goal does not unify because the the float is not bit-identical to the one in the call.

The question is, I though that Quintus went to printing floats in exponential notation to solve exactly this problem. Appended to this message is a copy of a message from David Bowen from April 1991, that makes precisely this claim.

This problems appears in version 3.2, running on Suns and SGIs.

John Dowding
dowding@ai.sri.com

Return-Path: quintus!quintus!dave@Sun.COM
Received: from drakes.ai.sri.com by Sunset.AI.SRI.COM (4.1/4.16)
id AA06175 for dowding; Tue, 2 Apr 91 11:49:52 PST
Received: from Sun.COM by drakes.ai.sri.com (4.1/4.16)
id AA13989 for dowding@ai.sri.com; Tue, 2 Apr 91 11:49:48 PST
Received: from sun.Eng.Sun.COM (sun-bb.Corp.Sun.COM) by Sun.COM (4.1/SMI-4.1)
id AA23348; Tue, 2 Apr 91 11:48:20 PST
Received: from quintus.UUCP by sun.Eng.Sun.COM (4.1/SMI-4.1)
id AA29540; Tue, 2 Apr 91 11:48:17 PST
Received: from odysseus.quintus.com by quintus.com; Tue, 2 Apr 91 11:13:57 pst
From: David Bowen <dave@quintus.com>
Date: Tue, 2 Apr 91 11:16:37 PST
Message-Id: <1311.9104021916@odysseus.quintus.com>
To: dowding@Sunset.AI.SRI.COM
Subject: Output of floating point numbers
Cc: hotline@quintus.com, ok@goanna.cs.rmit.OZ.AU

Thank you for your responses concerning my complaints about arithmetic. I was aware that you could force floating point numbers that correspond to integers to the integer form. However, I still think that the decision to print all floating-point numbers in exponent notation is a bad one.

The manual says (section G.1-1-2):

A floating point number (float) consists of a sequence of digits with an embedded decimal point, optionally preceded by a

minus sign, and optionally followed by an exponent consisting of upper- or lowercase 'E' and a signed base 10 integer.

This section is misleading, if it is in fact the case that the exponent is not optional, but is always produced by Prolog. This change certainly seems to be worth mentioning in the manual, and in the release notes.

Both exponential form and non-exponential form are allowed in the language. The Prolog system can use either and still be correct according to this paragraph.

This also leads to unexpected behaviour in other predicates, like:

| *?- number_chars(1.0, X).*

X = [49,46,48,69,43,48,48]

We could consider changing this, although it might break people's programs if they have come to rely on this behavior. I suppose we could have a prolog_flag to allow people to have it either way.

It also disturbs me that a portray clause:

portray(Float):- number(Float), write(Float).

mostly fixes this problem. This only fixes the printing of the floats, number_chars/2 still returns the exponent notation. Doesn't the Prolog top-level use write?

I don't see why this disturbs you if it gets the effect that you want! The Prolog top-level uses the equivalent of writeq, not write, to avoid ambiguities. (Its not exactly writeq because it goes through portray like print does.)

I know of no other programming language that will resort to printing exponent notation as the default. Most languages only rely on this notation when the number is either too large or too small to print conviently otherwise. It is just plain too hard to read.

Thanks for the comments John. I don't like this either, but let me explain the rationale as I remember it. (Richard O'Keefe was responsible for this decision.)

First, we want number_chars to do predictable things with floats. If you are going to do operations on the characters corresponding to the floating point number it may be helpful to have them in a consistent format.

Second, we want writeq to write out floats such that no precision is lost. That is, we want to be able to read back terms written by writeq and get exactly the same term.

Third, it would seem valuable to have number_chars and writeq be consistent. number_chars should also not lose any precision if you convert a float to a char-list and back.

Fourth, we use writeq to print the results of a top-level goal. If we didn't there could be nasty situations like distinguishing between a variable bound to the atom 'a+b' and one bound to the term a+b.

I guess the most debatable point is the first; would it really be a problem if number_chars attempted to use the minimal number of characters to represent the float without loss of precision (like the %g option in C)?

I will cc this to Richard to see if he would like to comment.

- **Next message:** David Martin: "Re: dynamic language models"
- **Previous message:** John Dowding: "problem with passing floats in OAA"

Re: dynamic language models

David Martin (*(no email)*)
Tue, 23 Jul 1996 20:43:31 +0000

- **Messages sorted by:** [date][thread][subject][author]
- **Next message:** John Dowding: "john makes a mistake"
- **Previous message:** John Dowding: "passing floating point numbers through library(tcp)"
- **Maybe in reply to:** John Dowding: "dynamic language models"

John -

It will be great to have a Gemini that accepts new lexical items at runtime.

I'm not sure if the current approach to this will meet your needs; some new design work may be needed.

Currently, I'm not aware of any dynamic changes made to SR grammars by agents (but this is something we've wanted, and something may already be in place).

With respect to the original NL agent, what happens is that the various agents globally post their lexical items on the facilitator's global data area (the blackboard). This is done by calling

write_bb(category, [LogicalPredicate, ListOfTokens]),

where ListOfTokens is in the format returned by QP library predicate read_sent:chars_to_words/2, and LogicalPredicate indicates what should be the result of recognizing the tokens in ListOfTokens.
If you go to
/homedir/oaa/agents/stable/
and do this:
grep write_bb\(*.pl | more

you'll see lots of examples of words being posted by different agents.

Then, the NL agent looks up these lexical items on the blackboard by calling read_bb/2. (So, the NL agent can construct a simple logical form using the logical predicates that are associated with the various lexical items.) To see how this is used, look in nl.pl, and search for calls to get_from_bb/2.

- Dave

- **Next message:** John Dowding: "john makes a mistake"
- **Previous message:** John Dowding: "passing floating point numbers through library(tcp)"
- **Maybe in reply to:** John Dowding: "dynamic language models"

john makes a mistake

John Dowding (*(no email)*)
Wed, 31 Jul 1996 18:36:51 -0700 (PDT)

- **Messages sorted by:** [date][thread][subject][author]
- **Next message:** Adam Cheyer: "StartIt: onready, ondisconnect"
- **Previous message:** David Martin: "Re: dynamic language models"

Suddenly, CommandTalk is completely out of control. It appears that every command I send is being executed *twice*, although my traces show everything is normal. The problem, which takes me 1/2 an hour to figure out, is that I have accidentally gotten two copies of the same agent running.

The sad part is, I made this same mistake about 6 months ago, and it didn't immediately occur to me what the problem was, although the symptoms are straight-forward enough.

My question is, could the OAA have saved me? Could we mark some agents as being *exclusive*, so that I'll get some kind of warning or failure when multiple agents post the same solvable to the facilitator.

- **Next message:** Adam Cheyer: "StartIt: onready, ondisconnect"
- **Previous message:** David Martin: "Re: dynamic language models"

StartIt: onready, ondisconnect

Adam Cheyer (*cheyer@ai.sri.com*)
Thu, 01 Aug 1996 17:26:08 -0700

- **Messages sorted by:** [date][thread][subject][author]
- **Next message:** Adam Cheyer: "Alert: XWindows agents"
- **Previous message:** John Dowding: "john makes a mistake"

All,

I added two new parameters that can be defined in a
StartIt configuration file for an agent:

onready <ICL> when the agent is ready (ie the little light
turns green), the ICL goal (or goals) is
broadcast using solve(ICLGoal, [broadcast])

ondisconnect <ICL> when an agent disconnects, either being killed
by StartIt or by dying in some other fashion,
the ICL goal (or goals) is broadcast using
solve(ICLGoal, [broadcast]).

<ICL> may be a single goal, ex:

onready inform_ui(db, ready)

or a list of goals, ex:

ondisconnect [inform_ui(db, disconnect),update(info)]

The rational for these two new features: there are times when agents
need to perform initialization handshaking with respect to one another.
If one agent is restarted for some reason, other agents might like to
know this information. Although each agent could setup individual
triggers to detect the coming and going of agents it is interested in,
in some cases it is much simpler and cleaner to put these project
level interactions directly in the startit config file, letting
StartIt to the work (it knows already who is connected and what their
state is).

Here's a more complete example:

```
#-----------------------------------------------------------------
appname Map & Position Database
oaaname db.root
appdir ${OAA_DEMO_DIR}
preline setenv HOST `hostname`
appline ${trace} ./db -oaa_name db robot/db_defs.pl
onready robot(initialize_position)
end
#-----------------------------------------------------------------
```

In the above example, a map database might contain coordinates for rooms and for moving objects (robots). If the database is restarted for some reason, all robots should be informed so that they can resend their initial positions in the map. In the above, all agents responding to the robot(initialize_position) solvable will receive this message when the map database agent is started or restarted.

The new startit executable has been compiled for sun4, and can be found in /home/trestle4/OAA/demo/startit.

I hope this will be of use to you...

-- Adam.

- **Next message:** Adam Cheyer: "Alert: XWindows agents"
- **Previous message:** John Dowding: "john makes a mistake"

Alert: XWindows agents

Adam Cheyer (*cheyer@ai.sri.com*)
Thu, 03 Oct 1996 11:44:48 -0700

- **Messages sorted by:** [date][thread][subject][author]
- **Next message:** Adam Cheyer: "Release Version"
- **Previous message:** Adam Cheyer: "StartIt: onready, ondisconnect"

After considerable effort, Harry Bratt tracked down
a bug that affects most C-XWindows agents when they
are run from the MWM window manager:

it turns out that some bug in MWM or WCL produces
spontaneous (unsolicited) calls to callback function
installed in many of our C agents with the call

XmpAddMwmCloseCallback(appShell, DeleteWindowCB, NULL);

Since the default DeleteWindowCB() callback generated by
the agent development tools calls app_done(), it may be that
app_done() is being called at random times during the execution
of your program, which can produce disasterous results in
certain cases (for example, the speech agent shuts down the speech
server).

THE FIX:

Please remove the call to XmpAddMwmCloseCallback in any of
your C-XWindows agents that use it, and delete the function
DeleteWindowCB(), which is really not necessary.

Thanks Harry!

-- Adam.

- **Next message:** Adam Cheyer: "Release Version"
- **Previous message:** Adam Cheyer: "StartIt: onready, ondisconnect"

Release Version

Adam Cheyer (*cheyer@ai.sri.com*)
Thu, 17 Oct 1996 08:05:42 -0700

- **Messages sorted by:** [date][thread][subject][author]
- **Next message:** Adam Cheyer: "Trigger Changes"
- **Previous message:** Adam Cheyer: "Alert: XWindows agents"

All,

As you know, we are working on coming out with a new, improved version of OAA which will be packaged in some form as "OAA-lite" and "OAA-pro". This new release will require some porting effort from all OAA developers, but better sooner than later, while there's less to port! Hopefully, you'll all agree the advantages are worth the effort.

Anyway, I thought I'd start sending out messages to y'all to keep you aware of what new changes will be coming, and to garner your suggestions as we battle with all of the various constraints to consider. We are trying to keep in mind all the requests we've seen from you over the years...

First, some general goals of the released version:

- A common API and functionality set across all released libraries!
- Documentation: Programmers guide, reference manual, and tutorial, all with _examples_ in multiple programming languages (documentation will favor Prolog & C, since most languages closely resemble one of these).
- Simplify, simplify...

As you probably know, a naming convention has been defined:

oaa_FunctionName() will be the format for oaa primitives in all languages.

This message is just a general introduction message. I shall start sending out msgs on more specific topics, soliciting your comments and suggestions.

Thanks,

Adam

- **Next message:** Adam Cheyer: "Trigger Changes"
- **Previous message:** Adam Cheyer: "Alert: XWindows agents"

Trigger Changes

Adam Cheyer (*cheyer@ai.sri.com*)
Thu, 17 Oct 1996 08:39:46 -0700

- **Messages sorted by:** [date][thread][subject][author]
- **Next message:** Adam Cheyer: "New event scheme"
- **Previous message:** Adam Cheyer: "Release Version"

Triggers will remain pretty much the same in the release
version of OAA, but there will be some syntactic differences
for installing triggers. In addition, there will be one or
two new features such as oaa_RemoveTrigger, and permanent triggers
(which live even after the agents who installed them die).
I'm also trying to figure out if there is a clean way to let
agents define NEW trigger types, using the ALARM trigger type
as an example.

Triggers evolved in OAA from add_trigger/4 to add_trigger/6, and
in the new version, guess what, they will go back to an
oaa_AddTrigger/4... but with parameters.

Here's the current proposal:

Declaration

Prolog: oaa_AddTrigger(+Type, +Condition, +Action, +Params)
C: void oaa_AddTrigger(char *Type, char *Condition, char *Action, char
*Params);

Description

Adds a trigger to either the local agent or a remote agent, as specified
by the Parameters.

The arguments are as follows:

Type: The trigger group: either event, data, test, or alarm.
Condition: A template used to select the trigger.
For data triggers, is in the form data(Item,Data).
For event triggers, the template matches the event.
For test triggers, is a solvable goal to try
For alarm triggers, represents a time expression
Action: An action which will be oaa_Interpreted when the trigger
fires.
oaa_Interpret will handle standard events (ev_halt, ev_trace_on),
OAA primitives (e.g. oaa_Solve()), and prolog built-ins (at
least on
Prolog agents). Also, an Action may be a complex request of more
than
one of these.
Params: May be an empty list or include some of the following
parameters:

local: Install the trigger on the local agent.
address(AgentNameOrID): Specifies the address of a remote agent on which
the trigger should be installed.
recurrence(R): R may be a positive integer representing the maximum number
of times the trigger should fire, or 'whenever', meaning the trigger
is in effect permanently. If the recurrence(R) option is not given,
the trigger will fire once and then be removed. Recurrence(if) and
recurrence(when) are kept for a sort of backward compatibility , but really
have no meaning since the trigger will be removed after firing anyway.
test(T): T contains a further test condition to oaa_Interpret before the
trigger will fire. T may be a complex expression.
on(Operation): The operation context:
for data triggers, Operation may be either write,
write_replace, retract, or replace. for event triggers,
may be send or receive.
Not used for alarm or test triggers.
lifespan(L): L = alive (default): trigger removed when agent quits or is killed
L = permanent: Trigger can live even after the agent quits.
L = time expression: Trigger is removed after a certain date/time.

If a destination is not explicitly specified in the parameter list, the default agent
on which to install the trigger is determined by the Type of the trigger:

Test triggers: the trigger will be installed on all agents with solvables
matching the trigger Template field.
Data triggers: the trigger will be installed on the Facilitator's global
data store.
Alarm triggers: alarm triggers are always installed on the alarm agent.
Event triggers: if no destination address is given, event triggers will be
installed on the local agent.

I think the use of parameters simplifies trigger installation, allows us to extend trigger
functionalities in the future, and removes the fact that some parameters are not used for
certain trigger types. For comparison, the current trigger scheme is:

add_trigger(Kind,Type,OpMask,Template,MoreConditions,Action)
where Kind = 'test,'data','alarm','event'

Type = when, if, or whenever (now extended by the
recurrence(R) parameter)
OpMask = on_retract, [on_send,on_receive], etc. (now the op(O)
parameter)
Template = Condition as defined above
MoreConditions = test parameter as defined above
Action

There were also separate primitives for add_local_trigger and
add_remote_trigger.

Other trigger-related primitives: oaa_RemoveTrigger/4,
oaa_CheckTriggers/3 (not usually
used by most agents since the agent library already calls this at the
appropriate places).

So, comments, ommisions, concerns, suggestions welcome!

Adam.

- **Next message:** Adam Cheyer: "New event scheme"
- **Previous message:** Adam Cheyer: "Release Version"

New event scheme

Adam Cheyer (*cheyer@ai.sri.com*)
Sun, 20 Oct 1996 17:34:25 -0700

- **Messages sorted by:** [date][thread][subject][author]
- **Next message:** Luc JULIA: "Re: New event scheme"
- **Previous message:** Adam Cheyer: "Trigger Changes"
- **Next in thread:** Luc JULIA: "Re: New event scheme"

Hi Everyone,

Here's my vision of what events will be like for the released version of OAA. Although this is what I have implemented so far, it is still malleable, so feel free to comment on what you like or don't!

==
SUMMARY of PROPOSAL

For those who don't have time to read:
the new version will add priorities to events, add flexible event queues, and change the default behavior for events received during a blocking solve.
==

If we are doing a blocking solve and waiting for the solutions to return, when some unexpected event arrives, what do we do? In the current OAA framework, we try to execute this event, because the event might be an important message, maybe even a message saying that we should abort our current polling for the event. If we deferred execution of this event until later, an abort-style message would be taken into account too late.

However, this can cause some problems, especially if the incoming event triggers work that requires another blocking solve. Now we have nested, recursive calls, and we are waiting for two different sets of solutions. Current OAA libraries have had some bugs related to this behavior (one problem regarding out-of-order return of solutions has been fixed in the Prolog agent library by the way, so grab the latest copy...)

So, the proposed fix is to add priorities to incoming events:

oaa_PostEvent(Event,Params)

can now take parameters :
address(Dest) : [Optional] Agent to send the event to
priority(P) : [Optional] Priority of event
(P: 1-10, 5=default, 10=high priority)
flush : [Optional] flush lower priority events from queue
(see description below)

Priority and flush are also added as parameters to oaa_Solve().

By default, if waiting for solutions in a blocking solve, unexpected events arriving will be queued for execution until AFTER the solve has completed, UNLESS the incoming event is of higher priority than the currently blocking solve, in which case it will be executed immediately.

Another problem that is fixed by this scheme is as follows:

Say requests come in to an agent faster than the work can be performed (for example, a text-to-speech agent might take several seconds for each request to complete). In this case, incoming events and solve requests will start backing up. Well, with priorities, you can send an event which will jump to the head of the line of executing events and be executed immediately. If a high-priority event is combined with the "flush" parameter, as the incoming event moves ahead of queued lower priority events, it removes them from the queue! Why would we want this? Well, in above example, we might want to indicate that all the backed up speech messages are no longer relevent, so the TTS agent should forget them, resynchronize to the current moment, and start saying things about the current state of the world.

So, priorities are useful not only during blocking solves, but as a way of detouring (and even clearing) a backup of events not yet handled.

So, as usual, comments or suggestions are always welcome!

Adam.

- **Next message:** Luc JULIA: "Re: New event scheme"
- **Previous message:** Adam Cheyer: "Trigger Changes"
- **Next in thread:** Luc JULIA: "Re: New event scheme"

Re: New event scheme

Luc JULIA (*(no email)*)
Sun, 20 Oct 1996 17:56:17 -0700 (PDT)

- **Messages sorted by:** [date][thread][subject][author]
- **Next message:** David Martin: "Re: New event scheme"
- **Previous message:** Adam Cheyer: "New event scheme"
- **Maybe in reply to:** Adam Cheyer: "New event scheme"
- **Next in thread:** David Martin: "Re: New event scheme"

Wow!!! Super!!!

Luc.

- **Next message:** David Martin: "Re: New event scheme"
- **Previous message:** Adam Cheyer: "New event scheme"
- **Maybe in reply to:** Adam Cheyer: "New event scheme"
- **Next in thread:** David Martin: "Re: New event scheme"

Re: New event scheme

David Martin (*(no email)*)
Mon, 21 Oct 1996 17:54:48 +0000

- **Messages sorted by:** [date][thread][subject][author]
- **Next message:** Adam Cheyer: "Re: New event scheme"
- **Previous message:** Luc JULIA: "Re: New event scheme"
- **Maybe in reply to:** Adam Cheyer: "New event scheme"
- **Next in thread:** Adam Cheyer: "Re: New event scheme"

Adam -

I like these proposals, they seem quite general, and they have obvious advantages over the current system.

But they do bring up one general concern, which is that an agent is extremely vulnerable to misguided or malicious control events arriving from other agents.
This the case not only for requests for flushing of queued events, but also for requests to add and remove triggers (as discussed in other messages).

Intuitively, it violates an agents "sovereignty" to have that agent automatically comply with all flush requests and trigger requests. And when OAA gets out into the real world, the vulnerability to misguided or malicious programming could quickly become a serious concern.

I don't have a clear answer to this. But perhaps something simple would go a long ways. Perhaps one of these:

(1) Each agent could specify from which other agents it will accept event flushing or trigger requests.

(2) When an event flush or trigger request arrives, it could be handled by a locally defined handler, if one has been defined. Such a handler would give an agent developer a place to write code to verify that it makes sense to comply with the arriving request. If the handler is not defined locally, then a default handler, provided by the agent library, could be used.

- Dave

Adam Cheyer wrote:
>
> *Hi Everyone,*
>
> *Here's my vision of what events will be like for the*
> *released version of OAA. Although this is what I have*
> *implemented so far, it is still malleable, so feel free*
> *to comment on what you like or don't!*
>
> ==

> *SUMMARY of PROPOSAL*
>
> *For those who don't have time to read:*
> *the new version will add priorities to events, add flexible*
> *event queues, and change the default behavior for events*
> *received during a blocking solve.*
> ==
>
> *If we are doing a blocking solve and waiting for the solutions*
> *to return, when some unexpected event arrives, what do we do?*
> *In the current OAA framework, we try to execute this event, because*
> *the event might be an important message, maybe even a message*
> *saying that we should abort our current polling for the event.*
> *If we deferred execution of this event until later, an abort-style*
> *message would be taken into account too late.*
>
> *However, this can cause some problems, especially if the incoming*
> *event triggers work that requires another blocking solve.*
> *Now we have nested, recursive calls, and we are waiting for*
> *two different sets of solutions. Current OAA libraries have*
> *had some bugs related to this behavior (one problem regarding*
> *out-of-order return of solutions has been fixed in the Prolog*
> *agent library by the way, so grab the latest copy...)*
>
> *So, the proposed fix is to add priorities to incoming events:*
>
> *oaa_PostEvent(Event,Params)*
>
> *can now take parameters :*
> *address(Dest) : [Optional] Agent to send the event to*
> *priority(P) : [Optional] Priority of event*
> *(P: 1-10, 5=default, 10=high priority)*
> *flush : [Optional] flush lower priority events from queue*
> *(see description below)*
>
> *Priority and flush are also added as parameters to oaa_Solve().*
>
> *By default, if waiting for solutions in a blocking solve, unexpected*
> *events arriving will be queued for execution until AFTER the solve*
> *has completed, UNLESS the incoming event is of higher priority than*
> *the currently blocking solve, in which case it will be executed*
> *immediately.*
>
> *Another problem that is fixed by this scheme is as follows:*
>
> *Say requests come in to an agent faster than the work can be*
> *performed (for example, a text-to-speech agent might take several*
> *seconds for each request to complete). In this case, incoming*
> *events and solve requests will start backing up. Well, with*
> *priorities, you can send an event which will jump to the*
> *head of the line of executing events and be executed immediately.*
> *If a high-priority event is combined with the "flush" parameter,*
> *as the incoming event moves ahead of queued lower priority events,*
> *it removes them from the queue! Why would we want this?*

> *Well, in above example, we might want to indicate that all the*
> *backed up speech messages are no longer relevent, so the TTS*
> *agent should forget them, resynchronize to the current moment,*
> *and start saying things about the current state of the world.*
>
> *So, priorities are useful not only during blocking solves, but as*
> *a way of detouring (and even clearing) a backup*
> *of events not yet handled.*
>
> *So, as usual, comments or suggestions are always welcome!*
>
> *Adam.*

- **Next message:** Adam Cheyer: "Re: New event scheme"
- **Previous message:** Luc JULIA: "Re: New event scheme"
- **Maybe in reply to:** Adam Cheyer: "New event scheme"
- **Next in thread:** Adam Cheyer: "Re: New event scheme"

Re: New event scheme

Adam Cheyer (*(no email)*)
Wed, 23 Oct 1996 16:28:49 -0700

- **Messages sorted by:** [date][thread][subject][author]
- **Next message:** Adam Cheyer: "Proposal: New Data management"
- **Previous message:** David Martin: "Re: New event scheme"
- **Maybe in reply to:** Adam Cheyer: "New event scheme"

SUMMARY OF MESSAGE:

- Security: Proposal to add a hook ("oaa_ValidateEvent") allowing
agent
library and agent programmer to check event before blindly accepting
to
execute it.
- Some discussion on Priorities of events and Threaded Messages
----

> *I like these proposals, they seem quite general, and they have obvious*
> *advantages over the current system.*
> *But they do bring up one general concern, which is that an agent is*
> *extremely vulnerable to misguided or malicious control events arriving*
> *from other agents.*
> *This the case not only for requests for flushing of queued events, but*
> *also for requests to add and remove triggers (as discussed in other*
> *messages).*
>
> *I don't have a clear answer to this. But perhaps something simple would*
> *go a long ways. Perhaps one of these:*
>
> *(1) Each agent could specify from which other agents it will accept*
> *event flushing or trigger requests.*
>
> *(2) When an event flush or trigger request arrives, it could be handled*
> *by a locally defined handler, if one has been defined. Such a handler*
> *would give an agent developer a place to write code to verify that it*
> *makes sense to comply with the arriving request. If the handler is not*
> *defined locally, then a default handler, provided by the agent library,*
> *could be used.*

David, you bring up an excellent but complicated point. Maliciousness is
not
contained to just flushing or trigger events -- MANY types of events
could do
harm or compromise security. Security is a very hard issue because it
involves:

-- Authentication of the sender, to be sure someone is not
maskerading as
another trusted agent.

-- Authentication of the message contents, to make sure someone
hasn't tampered
with it along the way.
-- Encryption, so others can't spy on your private data
-- Restriction of events to guaranteed harmless events.
-- etc.

It's clear that security is a vital issue, but I don't think it's one we
are
going to solve in this version of the research system.

However, I DO propose that we add a primitive
"oaa_ValidateEvent(InComingEvent, OkEvent)"
which will be immediately called on the arrival of event. This
predicate has the
option of refusing to accept the event entirely (by failing), or of
modifying the event
in some way to produce an acceptable version of it (e.g. by removing the
flush
parameter, by setting limits on the priority value, ...). Some checks
will be built
into the agent library (one day we can do event authentication, etc.)
and if the
event passes library-level tests, the event can be further tested at the
agent level
with the callback oaa_AppValidateEvent().

Keith had some good remarks about priorities:

> *I concur with David Martin's observations, and would extend that one*
> *further: who is it that defines the priority of any given event?*
> *An agent would have to know a priori to set a higher priority on an*
> *event that is a response to a blocking request issued by some other*
> *requesting agent. That is, both the agents have to agree on the set of*
> *priority values and what they mean, reducing the generality of the*
> *agents themselves somewhat. In effect, the proposal implements a small*
> *"sequencing tag" protocol (modulo the flush behavior).*

It is true that the proposed scheme is not extremely sophisticated;
rather it is intended
to be a little more general than the current scheme, and to address
several concrete
problems. In general, most events are not expected to have priorities,
but there may
be several specific reasons for having them. Perhaps we could remove
some confusion by setting
priorities in symbolic terms instead of numerical terms:

- default (5): blocking solves are not interrupted by other events
- interrupt_blocking (7): a higher priority event that will start
executing on an agent
even while it's waiting for the return of a blocking solve
- immediate (10): highest priority, should jump ahead of all other
events in the queue and

execute immediately
- not_busy (3): a low priority event that should not take up cpu time ahead of other tasks
but could be executed only when no other events are in the queue.

Erland, Keith and David had some comments about threaded message retrieval:

In the Java library, each blocking solve creates a new thread to wait for the answers.
One problem noted is that we don't have a unique ID associated with outgoing and
incoming messages, guarantee that they match: if two threads send unifyable requests,
some confusion could occur. One good proposal is that we add this ID to the solve
events to guarantee the match. Erland also proposed that the ID could be used as a
hash function to quickly match an incoming event to the correct thread.

- **Next message:** Adam Cheyer: "Proposal: New Data management"
- **Previous message:** David Martin: "Re: New event scheme"
- **Maybe in reply to:** Adam Cheyer: "New event scheme"

Proposal: New Data management

Adam Cheyer (*cheyer@ai.sri.com*)
Thu, 24 Oct 1996 10:15:42 -0700

- **Messages sorted by:** [date][thread][subject][author]
- **Next message:** David Martin: "Re: Proposal: New Data management"
- **Previous message:** Adam Cheyer: "Re: New event scheme"
- **Next in thread:** David Martin: "Re: Proposal: New Data management"

Summary of Message

- Replace bb predicates (read_bb, write_bb,...) and generic database agent (db.pl) by integrating data management into all agents through new db primitives in the agent library.
- Database management will closely parallel event management, with something like a solvable list kept by the Facilitator about data types for each agent, and a do_event-like data handler.
- Will support replication for simple collaborative activities
----

Now that we have had a chance to discuss triggers and events, the next subject on my list is Data Management.

As you know, the current OAA system uses read_bb, write_bb, etc. as a way of maintaining a global data store on the Facilitator. This is useful for several reasons:

- An agent can post information (for example, status information) to the Facilitator which can be queried by other agents using Solve, without the first agent needing to be directly queried (and interrupted) by a standard solvable call.
- Because of this, frequently accessed information can be made available to the community by posting it to the Facilitator, which reduces network overhead when accessing the info.
- After inspecting some information available as data on the Facilitator, an agent can install a data trigger requesting automatic notification if the value ever changes. This is much more efficient than constant polling using solvables.

The current OAA system makes use of a special generic database agent (db.pl) to hold supplementary information for agents. This database agent provides solvables such as db_declare, db_assert, etc., allowing agents to add, remove, change and query data predicates, as well as set data triggers on the information.

My proposal is to integrate the database primitives provided by the db agent into the agent library of all agents, and to remove the read_bb, write_bb primitives altogether. Agents could now keep local databases, or install data and data triggers on any remote agent including the Facilitator. In addition, the write_bb only allowed predicates with ONE argument to be maintained by the Facilitator

(such as user_name('Adam Cheyer')). The new system would work with predicates of any arity.

We would not add versions of the db agent's solvables as primitives in the agent library:

oaa_DbDeclare(DataPred, Arity, Params) where params could include:

address(A): by default, data is managed by local agent, but with
address(parent), data will be managed by Facilitator,
as was done using write_bb.
visibility(V): by default, visibility will be 'public': other agents
can access and query the data. If V is 'private',
the data is just for use by the local agent.
lifespan(L): if L = permanent, data is saved to file when agent
quits, and reinstated the next time agent restarts.
if L = temporary (default), data values die with the
agent.
If the data is installed on a Facilitator, when agent
quits, the data will be removed if temporary, and
kept if permanent (see Trigger Management msg).
single_value: any db_assert will overwrite any previous values
default is 'multiple_value'
no_duplicates: at most one copy of each value will be stored.
default is 'with_duplicates'
shareable: Predicate will use replication among to allow
collaboration among multiple participants (see below)
and others!

oaa_DbAssert(DataPred, Params)
oaa_DbRetract(DataPred, Params)
Plus more primitives for Loading, Saving, Simulation, etc.

Data can be queried using the standard oaa_Solve predicate, and perhaps also by an oaa_DbSolve, which first checks local database before sending out communication requests elsewhere.

Data management will come to parallel that of event management. In the same way that we have a solvable list of goals for each agent, db_declare will add to a data_manageable list of predicates for each agent. When a db_assert primitive is executed, the Facilitator will find the correct agent(s) who manage this data type and tell it to add the information to their data store. Data triggers will act in the same way.

In a similar way to how a oaa_AppDoEvent() handler is defined for each agent SOLVABLE, an oaa_AppDataChange() handler can be defined for each DECLARED data predicate. Let's use an example. Say an add_icon(X,Y,IconType) event has been defined for a map display agent. The do_event predicate will record the state change (that there is a new icon), and then visually update the screen to reflect the change. The old way of programming this might have been:

% given an event to add a new icon to the display,
% add the icon to the list of visible icons and then draw it.

```
do_event(_KS, add_icon(X,Y,IconType)) :-
assert(visible_icon(X,Y,IconType)),
draw_new_icon(X,Y,IconType).
```

Now we will use:

```
oaa_AppDoEvent(_KS, add_icon(X,Y,IconType)) :-
db_assert(visible_icon(X,Y,IconType)).

oaa_AppDataChange(visible_icon(X,Y,IconType), asserted) :-
draw_new_icon(X,Y,IconType).
```

db_assert will add the data to the local database, and then call the event handler for the change. What's the advantage? Now, if the visible_icon predicate is declared 'shareable', when the data predicate changes locally on one agent, other members of a collaborative conference will synchronously receive changes to the shared predicate, and the data handler will be called, drawing the new icon on everyone's screen. The actual means of propogating the changes can be hidden by the architecture, and be based on SCOOT or other collaborative techniques.

So, as usual, remarks, comments, suggestions and criticisms welcome!

- **Next message:** David Martin: "Re: Proposal: New Data management"
- **Previous message:** Adam Cheyer: "Re: New event scheme"
- **Next in thread:** David Martin: "Re: Proposal: New Data management"

Re: Proposal: New Data management

David Martin (*(no email)*)
Thu, 24 Oct 1996 18:26:45 +0000

- **Messages sorted by:** [date][thread][subject][author]
- **Next message:** Adam Cheyer: "PC News"
- **Previous message:** Adam Cheyer: "Proposal: New Data management"
- **Maybe in reply to:** Adam Cheyer: "Proposal: New Data management"

Adam -

\>
\> *My proposal is to integrate the database primitives provided by the*
\> *db agent into the agent library of all agents, and to remove the*
\> *read_bb, write_bb primitives altogether. Agents could now keep*
\> *local databases, or install data and data triggers on any remote agent*
\> *including the Facilitator. In addition, the write_bb only allowed*
\> *predicates with ONE argument to be maintained by the Facilitator*
\> *(such as user_name('Adam Cheyer')). The new system would work with*
\> *predicates of any arity.*
\>
\> *We would not add versions of the db agent's solvables as primitives*
\> *in the agent library:*
\>
\> *oaa_DbDeclare(DataPred, Arity, Params) where params could include:*
\>
\> *address(A): by default, data is managed by local agent, but with*
\> *address(parent), data will be managed by Facilitator,*
\> *as was done using write_bb.*
\> *visibility(V): by default, visibility will be 'public': other agents*
\> *can access and query the data. If V is 'private',*
\> *the data is just for use by the local agent.*

We may well want to expand the visibility options. I am thinking primarily that it will be useful to distinguish between read and write visibility. For example, most "system" data kept by the facilitator (like names and addresses and status of various agents) should probably be only readable. We don't want thing to get too complex, but how about having 3 values: private, readable, writable. (Writable of course means readable and writable.)

\> *lifespan(L): if L = permanent, data is saved to file when agent*
\> *quits, and reinstated the next time agent restarts.*
\> *if L = temporary (default), data values die with the*
\> *agent.*
\> *If the data is installed on a Facilitator, when agent*
\> *quits, the data will be removed if temporary, and*
\> *kept if permanent (see Trigger Management msg).*

\> *single_value: any db_assert will overwrite any previous values*
\> *default is 'multiple_value'*
\> *no_duplicates: at most one copy of each value will be stored.*

> *default is 'with_duplicates'*

Very minor point: For boolean parameters, instead of having 2 distinct values like single_value and multiple_value, why not have a single value with one arg. So, it would be single_value(true) or single_value(false). As a convenience, if desired, the true form could still be abbreviated as just single_value.

> *shareable: Predicate will use replication among to allow*
> *collaboration among multiple participants (see below)*
> *and others!*
>
> *oaa_DbAssert(DataPred, Params)*
> *oaa_DbRetract(DataPred, Params)*
> *Plus more primitives for Loading, Saving, Simulation, etc.*

This is all great. I can see how these (and other) params will be extremely useful.

>
> *Data can be queried using the standard oaa_Solve predicate,*
> *and perhaps also by an oaa_DbSolve, which first checks local database*
> *before sending out communication requests elsewhere.*
>
> *Data management will come to parallel that of event management.*
> *In the same way that we have a solvable list of goals for each agent,*
> *db_declare will add to a data_manageable list of predicates for each*
> *agent.*

I'm still debating whether it's best to treat solvables and data as separate things, as you are proposing, or as one unified sort of thing, as Prolog does. But if we treat them as separate, what's the best characterization of the difference? Procedure vs. data? If so, maybe we now know what to replace the term "solvables" with: we now have "procedures" and "data" declared for each agent.

> *When a db_assert primitive is executed, the Facilitator will*
> *find the correct agent(s) who manage this data type and tell it to*
> *add the information to their data store. Data triggers will act in*
> *the same way.*

> *In a similar way to how a oaa_AppDoEvent() handler is defined for*
> *each agent SOLVABLE, an oaa_AppDataChange() handler can be defined*
> *for each DECLARED data predicate. Let's use an example...*

First, since agent data capabilities have been redefined, I assume we need to revisit the definitions of data triggers. Until now, data triggers have only applied to the facilitator. They will probably generalize nicely to accomodate your new data proposals. But my main concern is, it's confusing to have both data triggers AND oaa_AppDataChange handlers. I think the oaa_AppDataChange handlers are probably subsumed by data triggers, so we just need to make sure data triggers are sufficiently general, rather than introducing a new category of handlers. Does this fit in with your thinking?

- Dave

- **Next message:** Adam Cheyer: "PC News"
- **Previous message:** Adam Cheyer: "Proposal: New Data management"
- **Maybe in reply to:** Adam Cheyer: "Proposal: New Data management"

PC News

Adam Cheyer (*cheyer@ai.sri.com*)
Tue, 05 Nov 1996 16:53:50 -0800

- **Messages sorted by:** [date][thread][subject][author]
- **Next message:** Adam Cheyer: "C Library: structured message proposal"
- **Previous message:** David Martin: "Re: Proposal: New Data management"

All,

Some news regarding OAA and the PC:

- We've received the PC version of Quintus Prolog, so we have now compiled most Prolog agents (at least those without foreign function calls) for Windows 95/NT. This includes the Facilitator, so it is now possible to run standalone systems on a single PC.

- I've updated agentlib.[ch] and tcp.[ch] to be ANSI C compatible and to compile under either unix or PC (MS Visual C++ or Borland C++). When compiling on a PC, compile with -D definition flags for IS_WINDOWS and either IS_BORLAND or IS_MSCPP. On unix -DIS_UNIX is defined by default. However, you should make sure to -I<agentlib directory> when you compile, even if agentlib.[ch] are found in the current directory.

That's all for now...

Adam.

- **Next message:** Adam Cheyer: "C Library: structured message proposal"
- **Previous message:** David Martin: "Re: Proposal: New Data management"

C Library: structured message proposal

Adam Cheyer (*cheyer@ai.sri.com*)
Thu, 22 May 1997 16:56:47 -0700

- **Messages sorted by:** [date][thread][subject][author]
- **Next message:** David Martin: "Re: C Library: structured message proposal"
- **Previous message:** Adam Cheyer: "PC News"
- **Next in thread:** Victor S. Abrash: "Re: C Library: structured message proposal"

As OAA v2.0 Prolog implementation is nearing completion, we are starting to look at the C, C++ libraries.

There have been many requests to move from OAA v1.0's string based data manipulation to a more structured approach. Please read and comment on this proposal...

String-based vs.Structure-based data manipulation in C library
==

String-based Construction

Prolog terms were constructed using a string representation of the data. Malloc and sprintf were frequently used to embed arguments into the correct the string. This could be inconvenient because you had to correctly calculate the size of the string buffer to receive the data. Alternatively, strings were constructed using an incremental process, using the Append() function; this is not very efficient, as the string is continually realloc'ed and recopied as the string increases in length.

The programmer also had to make sure that the resulting buffer contained a string in a valid Prolog syntax; details such as doubling special characters (e.g. ') had to be handled by the programmer.

Ex:

```
*answers = (char *) malloc(strlen(arg) + 35);
sprintf(*answers, "[func('%s')]", double_quotes(arg));
```

String-based Parsing

Parsing predicates were used to extract the desired information from an incoming message. This technique was somewhat inefficient because the string functions relied heavily on making new copies of the data, which had to be correctly allocated and deallocated

by the programmer.

The programmer also was required to worry about removing and undoubling quotes from incoming values.

Ex:

```
char *arg3 = NULL;
NthElt(args, 3, &arg3);
remove_quotes(arg3);
undouble_quotes(arg3);
use(arg3);

OAA_FREE(arg3);
```

Structured Messages
===================

It has been proposed that a structure-based approach be used in the C version of OAA 2.0. Instead of receiving arguments of type char * containing the message, we would receive arguments of type PObj *, where a PObj (Prolog Object) would represent the data in a structured format. PObj would be a union containing a type (var, int, atom, struct, list, uninitialized) and a datafield according to type.

"['adam', a(1,X)]"

would be represented as something like:

```
-------------
| Type: List |
| NumArgs: 2 | ----------------
| arg[0] -----------------------------------------> | Type: atom |
| arg[1] ----> --------------- | Val: "adam" |
------------- | Type: Struct | ----------------
| Func: "a" | ------------
| NumArgs: 2 | | Type: int |
| arg[0]-----------------------> | Val: 1 |
| arg[1] --> ------------- -------------
---------------| | Type: var |
| Val: X |
-------------
```

Access Functions

Access functions would allow the user to inspect or traverse individual elements in the data structure without resorting to excessive copying, allocation and deallocation.

Ex:

```
if (PRO_IsList(args)) {
```

```
for (i = 0; i < args->NumArgs; i++)
PRO_Print(args->arg[i]);
}
```

It would no longer be required to add/remove extraneous quotes to atoms as in the string-based approach.

Structure Construction

In some cases, the easiest way to create a new structure may still be to use strings. A string parser will be included for converting from string notation to structured representation. The inverse is also useful,
for backward compatibility; if DoEvent provides a structured event to execute, you can convert it to a string and reuse the old string-based parsing functions.

Ex:

```
PObj *query = NULL;
PObj *params = NULL;

query = PRO_ParseString("goal([1,2,3], abc, X)");
params = PRO_ParseString("[]");

OAA_Solve(query, params, &answers);

PRO_FREE(query);
PRO_FREE(params);
```

However, when not using the PRO_ParseString command to construct data structures, structures can be created using construction primitives.

```
PObj *query = NULL;
PObj *l;
PObj *a;
PObj *v;

l = PRO_NewList(PRO_NewInt(1), PRO_NewInt(2), PRO_NewInt(3), NULL);
a = PRO_NewAtom("abc");
v = PRO_NewVar("X");
query = PRO_NewStruct("goal", PRO_NewList(l, a, v));

OAA_Solve(query, PRO_NewList(NULL), &answers);

PRO_FREE(query);
```

For creation of lists when the length is not known at compile time, there will be a function:

```
PObj *a;

/* Dynamically create a list of N new ints */
a = malloc(sizeof(*PObj) * n);
for (i = 0; i < n; i++) {
a[i] = PRO_NewInt(i);
}

l = PRO_NewListFromArray(a, n);
```

List concatenation, and manipulation would also be supported.

Potential problem: deallocation can get tricky. Since we now permit multiple pointers to the same space (instead of creating a new copy each time), the programmer has to be extremely careful not to deallocate the same space multiple times.

What do you think? Will this be an improvement over the String-based approach? Are there better proposals?

Thanks for your response.

-- Adam.

- **Next message:** David Martin: "Re: C Library: structured message proposal"
- **Previous message:** Adam Cheyer: "PC News"
- **Next in thread:** Victor S. Abrash: "Re: C Library: structured message proposal"

Re: C Library: structured message proposal

David Martin (*(no email)*)
Thu, 22 May 1997 17:33:07 -0700 (PDT)

- **Messages sorted by:** [date][thread][subject][author]
- **Next message:** Victor S. Abrash: "Re: C Library: structured message proposal"
- **Previous message:** Adam Cheyer: "C Library: structured message proposal"

Here are 4 initial reactions:

(1) Looks just about right, overall.

(2) I think it would be much better to call the new data structure ICL object (ICLObj) instead of Prolog object. And of course the related accessors, etc. would be renamed accordingly.

(3)

> *For creation of lists when the length is not known at compile time,*
> *there will be a function:*
>
> *PObj *a;*
>
> */* Dynamically create a list of N new ints */*
> *a = malloc(sizeof(*PObj) * n);*
> *for (i = 0; i < n; i++) {*
> *a[i] = PRO_NewInt(i);*
> *}*
>
> *l = PRO_NewListFromArray(a, n);*

Yes, but even here, it appears that the final length has to be known when the List object is created. What I mean is, there should be constructors/ modifiers for adding elements to an existing List object, and removing elements, which are efficiently implemented. I'm thinking of a linked list implementation, as opposed to an array implementation, of the list elements. (If you covered this somewhere in the message, sorry, I've just skimmed.)

(4)

> *Potential problem: deallocation can get tricky. Since we now permit*
> *multiple pointers to the same space (instead of creating a new copy*
> *each time), the programmer has to be extremely careful not to deallocate*
> *the same space multiple times.*

Well, that's just part of working in the C world, not our fault. One very common technique is to include a ref_count (integer) in the data structure to help with bookkeeping.

- Dave

Next message: Victor S. Abrash: "Re: C Library: structured message proposal"
- **Previous message:** Adam Cheyer: "C Library: structured message proposal"

Re: C Library: structured message proposal

Victor S. Abrash (*victor@speech.sri.com*)
Thu, 22 May 1997 19:09:34 -0700

- **Messages sorted by:** [date][thread][subject][author]
- **Next message:** Martin Fong: "Re[2]: C Library: structured message proposal"
- **Previous message:** David Martin: "Re: C Library: structured message proposal"
- **Maybe in reply to:** Adam Cheyer: "C Library: structured message proposal"

Adam,

I think the structure approach would be great. Another advantage would be fewer memory leaks.

>
> *Potential problem: deallocation can get tricky. Since we now permit*
> *multiple pointers to the same space (instead of creating a new copy*
> *each time), the programmer has to be extremely careful not to deallocate*
> *the same space multiple times.*
>

Put an extra field in the structure, ie,

struct Pobj {

... regular stuff ...

int validity;
}

The validity field can be set to a magic cookie when the structure is valid, otherwise zero. When you deallocate an object, first set the validity field to zero before freeing memory. Now any other pointers will see that the memory they point to isn't a valid Pobj, and can report an error. Note that you can also re-use the validity field to indicate what type of object the structure holds.

Also, when you do
query = PRO_NewStruct("goal", PRO_NewList(l, a, v));
you should probably decide whether all objects are copied (in which case you should free l, a, & v) or just pointed to (in which case it's an error to free them). As long as it's consistent, we should be ok.

>
>
> *What do you think? Will this be an improvement over the String-based*
> *approach? Are there better proposals?*
>

This approach should make it easier to write robust programs. Without some kind of change, OAA is just not robust enough for real use.

Victor

- **Next message:** Martin Fong: "Re[2]: C Library: structured message proposal"
- **Previous message:** David Martin: "Re: C Library: structured message proposal"
- **Maybe in reply to:** Adam Cheyer: "C Library: structured message proposal"

Re[2]: C Library: structured message proposal

Martin Fong (*mwfong@std.sri.com*)
Thu, 22 May 1997 19:29:37 -0700

- **Messages sorted by:** [date][thread][subject][author]
- **Next message:** Keith Skinner: "Re[3]: C Library: structured message proposal"
- **Previous message:** Victor S. Abrash: "Re: C Library: structured message proposal"

Adam,

"Victor S. Abrash" <victor@speech.sri.com> wrote

> > *Potential problem: deallocation can get tricky. Since we now*
> > *permit multiple pointers to the same space (instead of creating a*
> > *new copy each time), the programmer has to be extremely careful*
> > *not to deallocate the same space multiple times.*

> *Put an extra field in the structure, ie,*

> *struct Pobj {*

> *... regular stuff ...*

> *int validity;*
> *}*

> *The validity field can be set to a magic cookie when the structure*
> *is valid, otherwise zero. When you deallocate an object, first set*
> *the validity field to zero before freeing memory. Now any other*
> *pointers will see that the memory they point to isn't a valid Pobj,*
> *and can report an error. Note that you can also re-use the*
> *validity field to indicate what type of object the structure holds.*

This approach may have problems when there multiple references to the same structure and you recursively "free" a structure that contains such a reference -- viz., you might inadvertently "invalidate" all other references.

> *Also, when you do*
> *query = PRO_NewStruct("goal", PRO_NewList(l, a, v));*
> *you should probably decide whether all objects are copied (in which*
> *case you should free l, a, & v) or just pointed to (in which case*
> *it's an error to free them). As long as it's consistent, we should*
> *be ok.*

One issue is whether the members of the new structure are immutable. If they need to be and aren't, then a shallow copy approach is problematic and you'll need to perform a deep copy. OTOH, if you add a "this-is-a-static-field" member to the structure and provide set-value accessors, then David's suggestion of using reference counts would be practical.

As far as David's suggestion that you use linked-lists instead of arrays, it strikes me that that depends on how likely the list will be modified -- writing, or even using, iterators can be a bit laborious if you do it often enough.

Cheers!

...Martin

- **Next message:** Keith Skinner: "Re[3]: C Library: structured message proposal"
- **Previous message:** Victor S. Abrash: "Re: C Library: structured message proposal"

Re[3]: C Library: structured message proposal

Keith Skinner (*skinner@std.sri.com*)
Tue, 27 May 97 09:51:08 PDT

- **Messages sorted by:** [date][thread][subject][author]
- **Next message:** Adam Cheyer: "oaa-users mailing list"
- **Previous message:** Martin Fong: "Re[2]: C Library: structured message proposal"

Adam,

"Victor S. Abrash" <victor@speech.sri.com> wrote

> > *Potential problem: deallocation can get tricky. Since we now*
> > *permit multiple pointers to the same space (instead of creating a*
> > *new copy each time), the programmer has to be extremely careful*
> > *not to deallocate the same space multiple times.*

> *Put an extra field in the structure, ie,*

> *struct Pobj {*

> *... regular stuff ...*

> *int validity;*
> *}*

> *The validity field can be set to a magic cookie when the structure*
> *is valid, otherwise zero. When you deallocate an object, first set*
> *the validity field to zero before freeing memory. Now any other*
> *pointers will see that the memory they point to isn't a valid Pobj,*
> *and can report an error. Note that you can also re-use the*
> *validity field to indicate what type of object the structure holds.*

I feel obligated to point out that this approach does not work: once the memory has been freed, any stale references to the structure should not reference a member of that structure, because even with a "magic number" you have no way of telling if the address is valid (e.g., processes can shrink as well as grow), nor can you determine the probability of the occurence of the "magic number" at that location (especially the probability of another object of the same type, and therefore same "magic number," but with different contents re-using those locations). This strategy inevitably leads to either memory leakage or, worse, incorrect and/or abortive program behavior and is not portable.

If you have the requirement for multiple references, you must first define how those references will be used:

If the model is that the structures are generated-once and are read-only, then you can use D. Martin's reference counting proposal (with appropiate modification for static declarations, as per M. Fong's suggestion).

If, however, you expect those references to be modified by any of the
program elements that make the references then you will also need the
ability to make deep copy references -- there are probably going to be
as many cases where the code making the reference wants the "modified"
reference as those who want the "original".

To be most general, support both deep-copy and reference-counting.
But again, this depends on how you want these structures to be used!

-Keith

- **Next message:** Adam Cheyer: "oaa-users mailing list"
- **Previous message:** Martin Fong: "Re[2]: C Library: structured message proposal"

oaa-users mailing list

Adam Cheyer (*cheyer@ai.sri.com*)
Mon, 19 Oct 1998 18:33:43 -0700

- **Messages sorted by:** [date][thread][subject][author]
- **Next message:** The Emminizer's: "OAA Mailing List"
- **Previous message:** Keith Skinner: "Re[3]: C Library: structured message proposal"

OAA-sters,

After a long period of inactivity, we are reviving the oaa-users mailing list. If you wish to be removed from the list, or you know of others who would like to be added, please let met know.

To send mail, send to oaa-users@ai.sri.com.

To browse the archive, see :

http://www.ai.sri.com/~oaa/distribution/MailArchive/oaa-users/

User login and password for the archive are:

login: oaa
password: sri-oaa!

Regards,

Adam.

```
--
Adam J. Cheyer                          Artificial Intelligence Center
Computer Scientist                      SRI International
Email: cheyer@AI.SRI.COM                Mail Stop: EJ217
WWW: http://www.ai.sri.com/~cheyer/     333 Ravenswood Avenue
Telephone: (650) 859-4119               Menlo Park, CA  94025-3493
Fax: (650) 859-3735                     USA
```

- **Next message:** The Emminizer's: "OAA Mailing List"
- **Previous message:** Keith Skinner: "Re[3]: C Library: structured message proposal"

OAA Mailing List

The Emminizer's (*emminize@erols.com*)
Fri, 23 Oct 1998 19:04:15 -0400

- **Messages sorted by:** [date][thread][subject][author]
- **Next message:** Adam Cheyer: "Strings in the OAA C library"
- **Previous message:** Adam Cheyer: "oaa-users mailing list"

Adam,

Please add me to your oaa-users mailing list.

Sorry I missed you last week in MD, but I was on a much-needed vacation. :-)

Thanks,

Paula Emminizer, SAIC

- **Next message:** Adam Cheyer: "Strings in the OAA C library"
- **Previous message:** Adam Cheyer: "oaa-users mailing list"

Strings in the OAA C library

Adam Cheyer (*cheyer@ai.sri.com*)
Fri, 30 Oct 1998 06:23:17 -0800

- **Messages sorted by:** [date][thread][subject][author]
- **Next message:** Ralph Becket: "Re: Strings in the OAA C library"
- **Previous message:** The Emminizer's: "OAA Mailing List"

Hi Andrew,

> > *2. I have been seeing some segmentation faults with some of our C*
> > *agents. The simpler ones that always return a constant string work,*
> > *but the ones which allocate and return a new C string often seg fault*
> > *right before sending the data on to the facilitator. This error is*
> > *cumulative and nondeterministic in that it usually doesn't happen*
> > *until after a few successful queries but it's pretty likely to happen*
> > *within 6 or 8 queries. Is there anything in particular about the*
> > *allocation of the string we pass off to OAA that we should know about*

I am a little confused by your question, particularly by the part that goes "always return a constant string" vs. "allocate and return a new C string". But I'll tell you some information about strings and allocation/deallocation and maybe this will shed some light on the subject.

In OAA, there's lots of string manipulation functions (OAA 2.0 will just pass you a structure that you can traverse with pointers, much nicer! No more double_quotes(), remove_quotes(), undouble_quotes(), etc.) for the ICL. Most ICL parsing functions return their result in a newly allocated string created using malloc(Size) or strdup(Str), and then you can dispose of them using the macro
OAA_FREE(result). OAA_FREE(ptr) just calls free(ptr), and then sets the ptr to be NULL. This is because we've noticed that strange things happen if you mistakenly free() the same pointer more than once, so OAA_FREE() guarantees this doesn't happen. This could be one source of your random error.

Example of good string use:

NthElt(args, 2, &result);
Use(result);
OAA_FREE(result);

One gotcha might come from the use of the Term() ICL parsing predicate, which splits a comma separated list (e.g. "a, b, c") into a first element and rest elements. Whereas ALMOST every ICL predicate returns the results in new space which should be free'd, the last argument of Term() returns a pointer back into the original string (first argument), and this shouldn't be OAA_FREE'd. Term() was written this way for the special purpose of iterating down lists, and really shouldn't be used for any other reason. Here's the "right" way to iterate down a list of

elements:

```
char *p, *list, *elt;

// The right way to iterate lists
list = strdup("e1, e(2), e(e(3))"); // list is in newly allocated
space
p = list; // p will iterate through list, one elt at a time
while (p && *p) { // Keep iterating until p is empty string
Term(p, &elt, &p); // NEVER free p!!! It points into list's
space.
Use(elt);
OAA_FREE(elt);
}
OAA_FREE(list); // Free the original list space
```

By the way, this way to iterate is exponentially more efficient than
using something
like:

```
// REALLY INEFFICIENT!
for (i = 1; i++; i <= list_len(list)) {
NthElt(list, i, &elt);
Use(elt);
OAA_FREE(elt);
}
```

This is because predicates like list_len() and NthElt() themselves must
iterate over
the list using Term().

A last few words on the do_event() function in a C agent program.
do_event() should
return a boolean value indicating whether or not to use the answers
returned in the
last argument. This argument returns a newly allocated string (using
malloc() or strdup())
containing a Prolog list of results matching the incoming query, or
strdup("[]")
(the empty list "[]" represents failure). NEVER just set the result to
a "constant" string:

```
*answers = "[]"; // BAD
```

Here's a sample do_event function for the query "a(X)" which returns the
solution a(1):

```
int do_event(char *KS, char *func, char *args, char **answers) {
int numArgs = list_len(args);
int processed = TRUE;
char arg1 = NULL;

// for solvable: a(X)
if ((strcomp(func, "a") == 0) && (numArgs == 1)) {
NthElt(args, 1, &arg1);
```

```
if (is_var(arg1) || (strcomp(arg1, "1") == 0) {
*answers = malloc(BIG_ENOUGH); // Can use malloc to create
space for answer
sprintf(*answers, "[a(%d)]", 1); // Stores result into answer space
}
else
*answers = strdup("[]"); // Can use strdup to return result
}
else processed = FALSE; // not returning a result for this query

return processed;
}
```

Hope this helps. If you are really stuck, maybe you can send over the
OAA wrapper part
of your code and we can take a quick look to see if anything stands out
as unusual.

Regards,
Adam.

--
Adam J. Cheyer
Computer Scientist
Email: cheyer@AI.SRI.COM
WWW: http://www.ai.sri.com/~cheyer/
Telephone: (650) 859-4119
Fax: (650) 859-3735

Artificial Intelligence Center
SRI International
Mail Stop: EJ217
333 Ravenswood Avenue
Menlo Park, CA 94025-3493
USA

- **Next message:** Ralph Becket: "Re: Strings in the OAA C library"
- **Previous message:** The Emminizer's: "OAA Mailing List"

Re: Strings in the OAA C library

Ralph Becket (*rwab1@cam.sri.com*)
Fri, 30 Oct 1998 15:26:49 +0000

- **Messages sorted by:** [date][thread][subject][author]
- **Next message:** Sharon Settnek: "Inquiry concerning NLP API"
- **Previous message:** Adam Cheyer: "Strings in the OAA C library"

This may or may not be of any use, but there is a conservative garbage collector for C that is freely available.

Here's a starting point: http://reality.sgi.com/boehm_mti/pubs.html

Conservative GC works by considering any number in the heap or on the stack that could be a pointer as if it really was a pointer. Blocks that are not referenced are freed. Apparently it has a surprisingly low impact on performance. The Mercury people used it to bootstrap their compiler and have been using it for two or three years now with no ill effects.

Cheers,

Ralph

```
--
Ralph Becket  |  rwab1@cam.sri.com  |  http://www.cam.sri.com/people/becket.html
```

- **Next message:** Sharon Settnek: "Inquiry concerning NLP API"
- **Previous message:** Adam Cheyer: "Strings in the OAA C library"

Inquiry concerning NLP API

Sharon Settnek (*sharons@teton.ncsc.mil*)
Tue, 03 Nov 1998 10:23:36 -0500

- **Messages sorted by:** [date][thread][subject][author]
- **Next message:** Adam Cheyer: "Re: Inquiry concerning NLP API"
- **Previous message:** Ralph Becket: "Re: Strings in the OAA C library"
- **Next in thread:** Adam Cheyer: "Re: Inquiry concerning NLP API"

I was just wondering if anyone had any ideas on how I could implement the
following agent:

I am writing an agent that attempts to "guess" what a user is asking if the
NLP can not answer the question. For example, if a user asks: "What is John Doe's identifier?" (a question that the NLP would/should not understand), I want to return a list of possible questions that try to guess what the user is asking - "What is John Doe's user id?, What is John Doe's telephone #" , etc. Once the list is generated, the user could choose what information they want. All understood questions

will be logged for analysis.

Is there any way I can directly communicate with the NLP (API?) to obtain
the sentence proper noun, nouns, and verbs? I know I can do a writeBB

to write to a bulletin board, and a readBB to read what I write, but, can I
do a readBB of what the NLP is writing (e.g. readBB of proper noun)? Is

the NLP writing anything to the bulletin board? I am not that familiar with
the capabilities of the NLP. Does anyone have any ideas on how I could implement this? Or, is there another way to do this other than using the
bulletin board?

Any help would be appreciated. Thanks.

- **Next message:** Adam Cheyer: "Re: Inquiry concerning NLP API"
- **Previous message:** Ralph Becket: "Re: Strings in the OAA C library"
- **Next in thread:** Adam Cheyer: "Re: Inquiry concerning NLP API"

Re: Inquiry concerning NLP API

Adam Cheyer (*(no email)*)
Tue, 03 Nov 1998 18:57:51 -0800

- **Messages sorted by:** [date][thread][subject][author]
- **Next message:** Stuart Lowry: "Loopy Triggers"
- **Previous message:** Sharon Settnek: "Inquiry concerning NLP API"
- **Maybe in reply to:** Sharon Settnek: "Inquiry concerning NLP API"

If I understand your question right, you would like to :

1. Log all requests to the natural language agent,
recording whether the english input was correctly parsed
or not.

2. If the NL agent is NOT capable of parsing the request
(i.e., "parse failed") be able to propose a list of other
questions that could be suggested to the user.

You should be able to write an agent which installs a trigger,
asking the facilitator to send it a notification message with the
result of every nl_to_icl() request sent to the NL agent. This can
be used to log the input questions and resulting solutions, and perhaps
to provide the "hook" to use the input sentence to suggest alternative
queries.

To set the trigger, create an agent which responds to the solvable
"nl_results(Sentence, Results)", and after connection to the
Facilitator, install the following trigger (written here in Java):

```
// Connects to Facilitator
agentBean.connect();

// Adds a trigger: whenever the facilitator receives the results of an
// nl_to_icl() query, sends a notification message using the nl_results
// solvable.
agentBean.lib.addRemoteTrigger("event", "whenever", "OnReceive",
"event(AgtId,
solved(GoalId,FromAgt,nl_to_icl(Sent,Parse,Params),P,Solutions))",
"true", "post_query(nl_results(Sent,Solutions),[broadcast])");
```

Everytime DCG_NL receives a request, your agent should receive a copy of
the results
DCG_NL returns, where you can log them in a file or whatever.

Another part of your question asked whether your agent can have access
to the vocabulary
used by dcg_nl. In the latest version of dcg_nl (which I will send you
in a following
message), there is a solvable:

nl_word(Type, Def)

where you can read the definitions for any word used by DCG_NL. "Type"
may be one
of the following:

noun
verb
imp_verb
inf_verb
intr_v
intr_imp_v
pn
adj
adv

To read all noun definitions, for example, use
oaa.lib.solve("nl_word(noun, Def)","[]");
Definitions are stored as lists : [WordsSolvable,
[atom(word1),atom(word2)]]
so to read the definition for "mail", or to see if the noun "mail" is
defined, you could
try oaa.lib.solve("nl_word(noun, [Solvable, [atom(noun)]])", "[]").

Hope this helps!

-- Adam.

- **Next message:** Stuart Lowry: "Loopy Triggers"
- **Previous message:** Sharon Settnek: "Inquiry concerning NLP API"
- **Maybe in reply to:** Sharon Settnek: "Inquiry concerning NLP API"

Loopy Triggers

Stuart Lowry (*sclowry@teton.ncsc.mil*)
Wed, 02 Dec 1998 15:20:07 -0500

I have a two part question regarding the "constraint" on the trigger API that I hope someone can help me with. I am writing an agent called 'trader' that sets a trigger to capture all events. I am filtering out my own events as the constraint to avoid a continuous loop. My problem is that there may be other agents that set Triggers to capture all the events like too (like a "monitor").

I am setting the constraint like the following:
constraint = "(KS\\== trader)";

question a/
How do I include more than 1 KS to constrian...I tried the following with no luck: constraint = "[(KS\\== trader),(KS \\== monitor)]"; but that did not work.

question b/
Fortunately my trader knew about an agent called "monitor" that was setting a similar trigger. Is there a rule of thumb that agents like these should advertise themselves as a type of "class" and that the similar agents can filter out events that originate from this similar class of agents?

Re: Loopy Triggers

Adam Cheyer (*(no email)*)
Thu, 03 Dec 1998 08:24:08 -0800

- **Messages sorted by:** [date][thread][subject][author]
- **Next message:** Stuart Lowry: "writeBB <-> readBB question"
- **Previous message:** Stuart Lowry: "Loopy Triggers"
- **Maybe in reply to:** Stuart Lowry: "Loopy Triggers"

Stuart Lowry wrote:
>
> *I have a two part question regarding the "constraint" on the trigger API*
> *that I hope someone can help me with. I am writing an agent called*
> *'trader' that sets a trigger to capture all events. I am filtering out*
> *my own events as the constraint to avoid a continuous loop. My problem*
> *is that there may be other agents that set Triggers to capture all the*
> *events like too (like a "monitor").*
>
> *I am setting the constraint like the following:*
> *constraint = "(KS\\== trader)";*
>
> *question a/*
> *How do I include more than 1 KS to constrian...I tried the following*
> *with no luck: constraint = "[(KS\\== trader),(KS \\== monitor)]"; but*
> *that did not work.*

You could write the constraint as

"(KS \\== trader, KS \\== monitor)"

> *question b/*
> *Fortunately my trader knew about an agent called "monitor" that was*
> *setting a similar trigger. Is there a rule of thumb that agents like*
> *these should advertise themselves as a type of "class" and that the*
> *similar agents can filter out events that originate from this similar*
> *class of agents?*

Good question!
This is a rather new problem that's been raised with the advent of the monitor program (the first agent that has really done much with event triggers),
and I'm not sure the best way of handling it yet. I'm open for suggestions....
My initial thought is that in OAA there should be a way to either explicitly or
implicitly send events which are not trapped by event triggers. For instance,
maybe every event produced as the result of a event trigger firing should NOT trigger
other event triggers. I think I prefer explicit programmatic control as well,
allowing a programmer to override the default of whether triggers fire

or not.

// Agent will receive an update_event() message every time
// any agent sends or receives any message (useful for logging).
// Note the parameter evt_triggers(false) so that the update_event
// message will not try to trace itself.
Java: oaa.lib.addRemoteTrigger("event", "whenever", // Type &
recurrence
"Op", "event(AgentId,Ev)", // Condition
"true", // No additional
test
"post_query(update_event(Op,AgentId,Ev),[broadcast,
evt_triggers(false)])");

Note, the above parameter (evt_triggers(false)) is not yet implemented,
just
a proposal.

Maybe such a scheme should include classes of events, or maybe
priorities
that allow higher-level requests to again override lower-level
preferences.

Any of these ideas will require modification to the agent library and
recompilation
of the Facilitator, but I'm certainly looking for comments or
proposals. I'd like
to incorporate something that handles this problem into a new version of
an OAA 1.0
facilitator so that we can test it, smooth out any problems, and then
incorporate
this in OAA 2.0.

-- Adam.

- **Next message:** Stuart Lowry: "writeBB <-> readBB question"
- **Previous message:** Stuart Lowry: "Loopy Triggers"
- **Maybe in reply to:** Stuart Lowry: "Loopy Triggers"

writeBB <-> readBB question

Stuart Lowry (*sclowry@teton.ncsc.mil*)
Wed, 16 Dec 1998 09:35:15 -0500

- **Messages sorted by:** [date][thread][subject][author]
- **Next message:** Adam Cheyer: "Re: writeBB <-> readBB question"
- **Previous message:** Adam Cheyer: "Re: Loopy Triggers"
- **Next in thread:** Adam Cheyer: "Re: writeBB <-> readBB question"

I was wondering if there is a slick way to do the following:
imagine that I am issuing facts to the BB like the following:

writeBB("noun",[emp, [emp,name,ssn,'employee id']]);
writeBB("noun",[phone, [phone,'phone number',telephone]]);

I would like issue a single readBB and get only the nouns that pertain
to the "meaning" item:

String phone_words = readBB("noun","[phone, X]");
and phone_words = "[phone,'phone number',telephone]";

as it stands I can only request ALL of the nouns and then I have to
traverse all the content to find the meaning that I want:
String phone_words = readBB("noun","X");
and phone_words = "readBB(noun,[emp,[emp,name,ssn,'employeeid']]),
readBB(noun,[phone, [phone,'phone number',telephone]])";

Thanks in advance,
Stuart Lowry

- **Next message:** Adam Cheyer: "Re: writeBB <-> readBB question"
- **Previous message:** Adam Cheyer: "Re: Loopy Triggers"
- **Next in thread:** Adam Cheyer: "Re: writeBB <-> readBB question"

Re: writeBB <-> readBB question

Adam Cheyer (*(no email)*)
Thu, 17 Dec 1998 08:52:16 -0800

- **Messages sorted by:** [date][thread][subject][author]
- **Next message:** John Dowding: "New facilitator prompting for port"
- **Previous message:** Stuart Lowry: "writeBB <-> readBB question"
- **Maybe in reply to:** Stuart Lowry: "writeBB <-> readBB question"

Stuart,

Any facts written to the Facilitator using writeBB() can also
be read using solve() instead of readBB(). For example, if
we do :

writeBB("noun","[emp, [emp,name,ssn,'employee id']]");
writeBB("noun", "[phone, [phone, telephone]]");

we can read the values using either

readBB("noun", "X")
or
solve("noun(X)", "[]")

When making these queries, unification is taken into account,
so you can do the kind of pattern matching you requested in your message.
For example, to get only those noun definitions that match the "meaning"
of "phone", you can do:

solve("noun([phone, X])", "[]").

-- Adam.

Stuart Lowry wrote:
\>
> *I was wondering if there is a slick way to do the following:*
> *imagine that I am issuing facts to the BB like the following:*
\>
> *writeBB("noun",[emp, [emp,name,ssn,'employee id']]);*
> *writeBB("noun",[phone, [phone,'phone number',telephone]]);*
\>
> *I would like issue a single readBB and get only the nouns that pertain*
> *to the "meaning" item:*
\>
> *String phone_words = readBB("noun","[phone, X]");*
> *and phone_words = "[phone,'phone number',telephone]";*
\>
> *as it stands I can only request ALL of the nouns and then I have to*
> *traverse all the content to find the meaning that I want:*
> *String phone_words = readBB("noun","X");*
> *and phone_words = "readBB(noun,[emp,[emp,name,ssn,'employeeid']]),*
> *readBB(noun,[phone, [phone,'phone number',telephone]])";*

\>
\> *Thanks in advance,*
\> *Stuart Lowry*

- **Next message:** John Dowding: "New facilitator prompting for port"
- **Previous message:** Stuart Lowry: "writeBB <-> readBB question"
- **Maybe in reply to:** Stuart Lowry: "writeBB <-> readBB question"

New facilitator prompting for port

John Dowding (*(no email)*)
Thu, 22 Apr 1999 10:33:36 -0700

- **Messages sorted by:** [date][thread][subject][author]
- **Next message:** John Dowding: "OAA library for Perl"
- **Previous message:** Adam Cheyer: "Re: writeBB <-> readBB question"

The new facilitor that I got from /home/trestle4/OAA has a new feature that I would like to disable.

When I start a new facilitator on a host/port that already has one running, it used to just die silently (which was fine). Now it prompts:

Exception reported:
tcp_mishap(tcp_listen.bind,48)

Currently unable to access gansett port 5082.
Try again? [y)es, n)o, c)hange, h)elp]:

and I have to respond 'no' all the time.

Can I get the old behavior back?

- **Next message:** John Dowding: "OAA library for Perl"
- **Previous message:** Adam Cheyer: "Re: writeBB <-> readBB question"

OAA library for Perl

John Dowding (*(no email)*)
Tue, 06 Jul 1999 18:35:36 -0700

- **Messages sorted by:** [date][thread][subject][author]
- **Next message:** Luc JULIA: "Re: OAA library for Perl"
- **Previous message:** John Dowding: "New facilitator prompting for port"
- **Next in thread:** Luc JULIA: "Re: OAA library for Perl"

I am considering writing an OAA library for Perl, and I was wondering if anyone had already done this.

My current plan is to just build on the C layer, rather than writing a new native library in Perl. I expect this would not be too hard, about a day's work.

- **Next message:** Luc JULIA: "Re: OAA library for Perl"
- **Previous message:** John Dowding: "New facilitator prompting for port"
- **Next in thread:** Luc JULIA: "Re: OAA library for Perl"

Re: OAA library for Perl

Luc JULIA (*Luc.Julia@sri.com*)
Tue, 06 Jul 1999 20:09:49 -0700

- **Messages sorted by:** [date][thread][subject][author]
- **Next message:** Stuart Lowry: "Large data sets"
- **Previous message:** John Dowding: "OAA library for Perl"
- **Maybe in reply to:** John Dowding: "OAA library for Perl"

John,

> *I am considering writing an OAA library for Perl, and I was wondering if*
> *anyone had already done this.*
>
> *My current plan is to just build on the C layer, rather than writing*
> *a new native library in Perl. I expect this would not be too hard,*
> *about a day's work.*

We actually wrote something very similar in Java, for awk. With a minimum amount of work, I guess that this java wrapper could call perl scripts as well.

Luc.

- **Next message:** Stuart Lowry: "Large data sets"
- **Previous message:** John Dowding: "OAA library for Perl"
- **Maybe in reply to:** John Dowding: "OAA library for Perl"

Large data sets

Stuart Lowry (*slowry@c3i.saic.com*)
Wed, 7 Jul 1999 08:40:44 -0400 (EDT)

- **Messages sorted by:** [date][thread][subject][author]
- **Next message:** Adam Cheyer: "Re: Large data sets"
- **Previous message:** Luc JULIA: "Re: OAA library for Perl"
- **Next in thread:** Adam Cheyer: "Re: Large data sets"

Does anyone have any suggestions for sending large data sets..
(1-10 M) around a distributed agent system? I tried sending the data
through the facilitator directly but my system bogged down. This could be
due to something else so I will try it again. I was also considering
adding a simple web server agent that essentially serves up local files.
This way I could pass a reference (URL) to my remote agents and they could
in turn fetch the data.

Suggestions?

Thanks in Advance.
Stuart

```
--
_____________________________________________________________________________
Stuart Lowry                                          stuart.lowry@cpmx.saic.com
```

- **Next message:** Adam Cheyer: "Re: Large data sets"
- **Previous message:** Luc JULIA: "Re: OAA library for Perl"
- **Next in thread:** Adam Cheyer: "Re: Large data sets"

Re: Large data sets

Adam Cheyer (*(no email)*)
Thu, 26 Aug 1999 17:01:01 -0700

- **Messages sorted by:** [date][thread][subject][author]
- **Previous message:** Stuart Lowry: "Large data sets"
- **Maybe in reply to:** Stuart Lowry: "Large data sets"

Didier is in Australia this week, but when he gets back, maybe he can talk about the peer-to-peer data stuff he did for the Java and C libraries in OAA 1.0. The Facilitator is used to help setup the connections, and then agents ship data back and forth directly.

-- Adam.

- **Previous message:** Stuart Lowry: "Large data sets"
- **Maybe in reply to:** Stuart Lowry: "Large data sets"

Exhibit 17

EXHIBIT 6

1.0
Distribution

Distribution Information

Documentation

Download

Troubleshooting

Contact & Community

OAA Homepage

SRI AI Center

The Open Agent Architecture™ 1.0 Distribution

Status: Unsupported Partial Release

About this distribution

This distribution is intended for developers who need to work with OAA 1.0 in order to support some existing system, or reuse existing agents based on OAA 1. Others should work with OAA 2.x. Please note that OAA 1.0 is not thoroughly documented and not well supported.

For those who need to reuse an OAA 1.0 agent, note that it is possible to use a mix of OAA 1.0 and OAA 2.x agents with the OAA 2.0 facilitator.

This OAA 1.0 distribution is released to the public under a non-exclusive end-user software license agreement.

Mailing List!

All users of OAA are **strongly encouraged** to subscribe to, and make use of, the oaa-users mailing list. There are no strings attached; your email address will not be used for any other purpose than the mailing list. Traffic is relatively light, but many useful questions get asked and answered on the list.

To **subscribe** to the mailing list, please send mail to oaa-users-request@ai.sri.com, with "subscribe" in the Subject. You will receive an acknowledgement of your request. Additional information, including unsubscribe instructions, are available on the Contact & Community page.

Release Notes

- **November 15, 1999**: Added OAA 1.0 library for Perl, along with simple test agent and basic documentation.
- **July 18, 1999**: Updated Runtime Distribution: several bugs fixed in Java library (oaa.jar), improvements made (both functional and bug-fixes) to Start-It and Monitor tools. Added new WebL script for Yahoo weather, because Intellicast weather changed their site.
- **February 3, 1999**: Documentation for the Java library of OAA 1.0

(javadoc).

- **January 8, 1999**: New version of the Lisp library, which includes alpha-level support for Allegro/NT.
- **November 20, 1998**: First release. Distribution for UNIX and PC, includes Runtime Distribution (Facilitator; Tools: Monitor, Start-It, Debug; WebL and DCG-NL oriented sample application). Agent libraries for many languages, with sample agent. See the new Tutorial (still a little rough...).
